UNIVERSITY of ALASKA ANCHORAGE

State of Alaska Election Security Project *Phase 1 Report*

Supplemental Documents

Prepared for Lieutenant Governor Sean Parnell and the State of Alaska Division of Elections

December 14th, 2007

These documents were reprinted with permission from the authors

Contents

Principal Investigator's Statement on Protection of Security-Sensitive Information

David Wagner
Principal Investigator
University of California, Berkeley

August 2, 2007

In May 2007, the California Secretary of State commissioned the University of California to conduct a security evaluation of the source code of several voting systems, as part of the Secretary's top-to-bottom review of these voting systems. As we prepared the final reports describing the findings from our source code review, we took great pains to ensure that our public reports would not inadvertently endanger the security of the country's elections. This note describes the approach we took.

In publicly discussing the security of any fielded system, there is often an inherent tension between our dual obligations as responsible engineers to precisely identify potential weaknesses and also to avoid aiding those who might seek to exploit them. These obligations themselves are closely intertwined, ultimately serving the common goal of making our society less vulnerable to the misconduct of criminals. Flaws cannot be fixed if they are not properly understood, and the modern history of technology repeatedly reminds us that we rely on the presumed ignorance of attackers only at great peril.

A common, widely accepted practice in the security literature is to describe attacks in sufficient detail to allow others to independently reproduce and evaluate the threat and, ultimately, build systems that better resist attack. Because of the severity of the attacks we found, and because we wanted to avoid making it easy for would-be attackers to subvert elections, we did not follow that practice here.

Instead, in preparing our public reports, we deliberately chose to err on the side of caution. We carefully screened all of the information that we included in our public reports. Our objective was to avoid reducing the amount of access an attacker would require to attack elections. We attempted to accomplish this by omitting details that would have the effect of converting an attack that would require reverse engineering or access to the source code into one that would not. These details were relegated to a confidential appendix provided to the Secretary of State. In some cases we deviated from this guideline when an attack scenario was already readily obvious from the interfaces presented to the user or from the previously published literature.

Our final reports are not intended to provide the level of technical detail that is typically expected in scientific publications. Reproducibility is at the heart of the scientific enterprise, and scientific standards generally dictate that technical data be fully disclosed to enable other scientists to independently reproduce the results. However, because we specifically wanted to avoid making it easy for would-be attackers to reproduce these attack scenarios in actual elections, our public reports omit certain technical information that would otherwise be expected under scientific norms. We recognize that independent scientists who seek to reproduce our results with the aim of building more robust systems may find the omissions frustrating.

In the end, our reports strike a careful balance between the public's interest in transparency into whether their voting systems are secure and the public's interest in being protected against the risks due to the disclosure of those flaws. We hope that future voting systems, better engineered than today's systems, will eliminate the need for such trade-offs.

Source Code Review of the Diebold Voting System

Joseph A. Calandrino
Princeton University[1]

Ariel J. Feldman
Princeton University

J. Alex Halderman
Princeton University

David Wagner[2]
U.C., Berkeley

Harlan Yu
Princeton University

William P. Zeller
Princeton University

July 20, 2007

This report was prepared by the University of California, Berkeley under contract to the California Secretary of State as part of a "Top-to-Bottom" review of electronic voting systems certified for use in the State of California.

[1]All author affiliations are for identification only.
[2]Team leader.

Executive Summary

This report is a security analysis of the Diebold voting system, which consists primarily of the AccuVote-TSX (AV-TSX) DRE, the AccuVote-OS (AV-OS) optical scanner, and the GEMS election management system. It is based on a study of the system's source code that we conducted at the request of the California Secretary of State as part of a "top-to-bottom" review of California voting systems.

Our analysis shows that the technological controls in the Diebold software do not provide sufficient security to guarantee a trustworthy election. The software contains serious design flaws that have led directly to specific vulnerabilities that attackers could exploit to affect election outcomes. These vulnerabilities include:

- **Vulnerability to malicious software**
  The Diebold software contains vulnerabilities that could allow an attacker to install malicious software on voting machines or on the election management system. Malicious software could cause votes to be recorded incorrectly or to be miscounted, possibly altering election results. It could also prevent voting machines from accepting votes, potentially causing long lines or disenfranchising voters.

- **Susceptibility to viruses**
  The Diebold system is susceptible to computer viruses that propagate from voting machine to voting machine and between voting machines and the election management system. A virus could allow an attacker who only had access to a few machines or memory cards, or possibly to only one, to spread malicious software to most, if not all, of a county's voting machines. Thus, large-scale election fraud in the Diebold system does not necessarily require physical access to a large number of voting machines.

- **Failure to protect ballot secrecy**
  Both the electronic and paper records of the Diebold AV-TSX contain enough information to compromise the secrecy of the ballot. The AV-TSX records votes in the order in which they are cast, and it records the time that each vote is cast. As a result, it is possible for election workers who have access to the electronic or paper records and who have observed the order in which individuals have cast their ballots to discover how those individuals voted. Moreover, even if this vulnerability is never exploited, the fact that the AV-TSX makes it possible for officials to determine how individuals voted may be detrimental to voter confidence and participation.

- **Vulnerability to malicious insiders**
  The Diebold system lacks adequate controls to ensure that county workers with access to the GEMS central election management system do not exceed their authority. Anyone with access to a county's GEMS server could tamper with ballot definitions or election results and could also introduce malicious software into the GEMS server itself or into the county's voting machines.

Although we present several previously unpublished vulnerabilities, many of the weaknesses that we describe were first identified in previous studies of the Diebold system (e. g., [26], [17], [18],

[19], [33], [23], and [14]). Our report confirms that many of the most serious flaws that these studies uncovered have not been fixed in the versions of the software that we studied.

Since many of the vulnerabilities in the Diebold system result from deep architectural flaws, fixing individual defects piecemeal without addressing their underlying causes is unlikely to render the system secure. Systems that are architecturally unsound tend to exhibit "weakness-in-depth" — even as known flaws in them are fixed, new ones tend to be discovered. In this sense, the Diebold software is fragile.

Due to these shortcomings, the security of elections conducted with the Diebold system depends almost entirely on the effectiveness of election procedures. Improvements to existing procedures may mitigate some threats in part, but others would be difficult, if not impossible, to remedy procedurally. Consequently, we conclude that the safest way to repair the Diebold system is to reengineer it so that it is secure by design.

Table of Contents

List of Issues

CHAPTER 1

Introduction

For years, many computer security researchers have been calling for state governments to conduct thorough, independent security studies of their electronic voting equipment. California was among the first states to do so, and we thank the Secretary of State for commissioning this study and for providing top-to-bottom access to the source code under review. We feel honored to have had the opportunity to participate.

Our analysis shows that the Diebold software we studied contains serious design flaws that have led directly to specific vulnerabilities, which attackers could exploit to affect election outcomes. By breaking the seal on just one voting machine, a criminal could launch a vote-stealing virus that could spread to every machine in a county. By manipulating a voting machine for a few minutes after the election, a corrupt volunteer poll worker could determine how each person who used that machine voted. These and many other attacks are feasible.

Furthermore, because the Diebold system suffers from systemic flaws, not just implementation defects, we cannot conclude that the list of vulnerabilities that we present in this report is exhaustive. Indeed, even as we write, we are uncovering new vulnerabilities and learning of additional vulnerabilities being identified by others [24]. Systems with architectural weaknesses tend to be fragile — even as known flaws are fixed, new ones tend to come to light.

Part of the promise of electronic voting is that technological and procedural safeguards can be combined to conduct elections more securely than ever before. The Diebold system does not live up to this promise, however, because its vulnerabilities allow a reasonably sophisticated attacker to surmount almost every technological barrier that is in place. As a result, the security of elections conducted on the Diebold system depend almost entirely on the effectiveness of election procedures.

Leaving the Diebold software largely unchanged and relying on procedural changes to mitigate the threats that we describe may seem like an appealing option, but we consider this to be a risky approach. First, although procedural changes are valuable, we are not confident that they can be completely effective. There are some vulnerabilities that are difficult, if not impossible, to mitigate procedurally. Second, because the Diebold system is vulnerable in so many ways, the procedures required to protect it would likely be extensive, complex, and hard to follow. Hence, we worry that despite the best efforts and intentions of election officials, the procedures would not be followed perfectly every time and the system would sometimes be left open to attack.

The severity of the design flaws in the Diebold system and our lack of confidence in the ability of changes in election procedures to compensate for them leads us to conclude that the surest way to repair the system is to redesign it.

About this Report This report was prepared by the University of California, Berkeley at the request of the California Secretary of State, as part of a "top-to-bottom" review of the state's electronic voting systems. This document is the final report of the team that examined the Diebold voting system source code.

The Diebold system source code review team was located at Princeton University and consisted of the six authors of this report. We frequently consulted with the Cleveland State University-based

document review team[1] and the UC Davis-based "Red Team"[2]. All opinions expressed in this report, however, are solely those of the authors.

We started work on May 31, 2007 and received the Diebold system source code on June 8, 2007. Work ended on July 20, 2007 with the delivery of this report, at which time we destroyed all proprietary materials.

Organization This report is organized as follows. The remainder of this chapter introduces the scope of our study, the methods we used, and the limitations of our findings. Chapter 2 describes the overall architecture and individual components of the Diebold election system as it is typically deployed. In Chapter 3, we highlight the most serious attack scenarios we found and discuss their implications. We identify high-level design and architectural problems in Chapter 4, followed by specific vulnerabilities in the individual components in Chapter 5. In Chapter 6, we discuss technical and procedural approaches to improving the security of the Diebold system. We conclude our report in Chapter 7. Appendix A outlines a generic threat model for large electronic voting systems such as the Diebold system. Appendix B contains additional details about the problems we found as well as source code excerpts; since it may contain vendor-proprietary information, this appendix has been designated "private."

1.1 System Overview

The Diebold software we reviewed is part of a system that includes touchscreen direct recording electronic (DRE) voting machines and optical scan voting machines for use at polling places as well as election definition, management, and counting software and hardware for use at a county election headquarters. The specific software components that we reviewed were:

- The GEMS 1.18.24.0 election management system
- The AccuVote-TSX DRE, including:
  - BallotStation version 4.6.4
  - Bootloader version BLR 7-1.2.1 and "Wildcat" platform
- The AccuVote-OS Precinct Count optical scan machine, version 1.96.6
- The AccuVote-OS Central Count optical scan machine, version 2.0.11.4[3]
- Vote Card Encoder, version 1.3.2
- Key Card Tool, version 4.6.1
- VC Programmer, version 4.6.1

In total, the reviewed software comprises about 300,000 source lines of code (SLOC) written in a variety of programming languages, including C, C++, and assembly language. (See Table 1.1.)

[1] Candice Hoke (team leader), Dave Kettyle, and Tom Ryan

[2] Robert P. Abbott (team leader), Mark Davis, Joseph Edmonds, Luke Florer, Elliot Proebstel, Brian Porter, Sujeet Shenoi, and Jacob Stauffer

[3] The state certification of the Diebold voting system identifies version 2.0.12 of the AccuVote-OS Central Count software, and that is the version that the Red Team was given. However, based on directory names and the contents of the source code, we appear to have been given the source code for version 2.0.11.4. We cannot account for the discrepancy between these versions and we have no way of knowing the impact it might have on our findings.

Component	*SLOC*	*Language(s)*
AV-OS Central Count 2.0.11.4	24K	(asm, C, C++)
AV-OS Precinct Count 1.96.6	20K	(asm, C)
AV-TSX Ballot Station 4.6.4	65K	(C++)
AV-TSX bootloader and "Wildcat"	71K	(asm, C, C++)
GEMS 1.18.24.0	116K	(C++)
Key Card Tool 4.6.1	1K	(C++)
Voter Card Encoder 1.3.2	1K	(C)
VCProgrammer 4.6.1	2K	(C++)
(total)	300K	

Table 1.1: The number of non-blank, non-comment source lines of code in each voting system component, as counted by David Wheeler's `sloccount` 2.26. All numbers have been rounded to the nearest thousand lines of code.

1.2 Methodology

Discovery of programming errors is a notoriously difficult problem in computer science, and no general methodology exists that is guaranteed to find all problems in even very small programs. Line-by-line source code analysis is an extremely time-consuming and laborious endeavor. Consequently, given the size of our team and the time we were allotted, it was unrealistic for us to attempt to examine every line of code or to find every defect in the Diebold system.

Instead, we focused our attention on the portions of the code that were most likely to have an impact on security and reliability. We also used the Fortify static analysis tool[4] to identify potential problem areas that warranted further manual investigation. We made no attempt to catalog all defects that might enable any particular kind of attack. Once we found several related vulnerabilities in the same portion of the code, we stopped looking for other vulnerabilities of the same type. Such an analysis is by its nature incomplete, and is particularly unlikely to discover deliberately introduced and obfuscated flaws, such as hidden "back doors" incorporated into the software by a malicious programmer.

Our focus was on whether the software contains effective safeguards against error and abuse aimed at altering election results, denying service, and compromising ballot secrecy. We also placed a special emphasis on systemic issues that might go beyond individual vulnerabilities. In general, our review attempted to explore questions such as:

- What are the trusted components of the system, when are they trusted and for what purposes? What parties are trusted and for what purposes? What are the implications of compromise of trusted components?
- Is the cryptography and key management sound? Is cryptography correctly used to protect sensitive data on untrusted media? Does the cryptography employ standard algorithms and protocols? Are keys managed according to good practices?
- Are security failures likely to be detected? Are audit mechanisms reliable and tamper-resistant? Is data that might be subject to tampering properly validated and authenticated?
- Can an untrusted or minimally trusted user escalate his capabilities beyond those for which he was authorized?
- Does the design and implementation follow sound, generally accepted engineering practices? Is code defensively written against bad data, errors in other modules, changes in environment, and so on?

[4] We are grateful to Fortify Software for making the tool available to us for this project at no charge.

- Is the system designed in a way that allows meaningful analysis? Is the architecture and code amenable to an external review (such as ours)? Could code analysis tools be usefully applied? Is there evidence of previous testing and other quality control practices?

1.3 Limitations of this Report

This report is an analysis of the source code to a particular version of the Diebold voting system and its conclusions do not necessarily apply to other versions of the Diebold system. It is also not intended to be an analysis of the security or reliability of the Diebold hardware.

Our analysis is based on the source code and documentation that we received from the State of California, Diebold, and the Independent Testing Authorities (ITAs). We made no attempt to validate the materials provided to us, nor do we have any way of knowing whether the source code provided to us matches the software that is actually used on election day. Any omissions or inaccuracies in the materials that were provided to us could have led to inaccuracies in this report.

Although we do not have a complete list of the Diebold voting system software, we are aware of several software components that the system uses that we did not receive. They are:

- JResultsClient
- The firmware for the AccuView Printer Module
- The Windows CE operating system used on the AccuVote-TSX
- The Windows operating system and other applications used on GEMS PCs
- Third-party libraries, such as the standard C libraries provided by the compiler vendor

Some of this software, such as the standard C libraries, may be classified as unmodified COTS (commercial off-the-shelf) software under the federal voting standards and thus may be exempt from disclosure to testing labs. Other software, such as JResultsClient, was apparently written by Diebold but was still not made available to us. In the absence of source code to the Windows CE operating system used on the AV-TSX, we were not able to verify whether it qualifies as unmodified COTS under the provisions of the federal voting standards.

This report is not intended to be a comprehensive evaluation of California election procedures. Nevertheless, in the course of our analysis, we discuss the extent to which various election procedures are able to mitigate security vulnerabilities in the Diebold software.

CHAPTER 2

Architecture

In this chapter we provide a high-level overview of the components of the Diebold system and describe how they are used in a typical deployment, as illustrated in Figure 2.1.

2.1 Components at Polling Places

There are several components that might be found at polling places, depending on county election practices:

- The *AccuVote-OS (AV-OS) Precinct Count* is an optical scan voting machine. During an election, voters mark paper ballots and feed them into the AV-OS. The AV-OS scans their ballot, interprets the marked votes, increments running counts of the number of votes for each candidate, and deposits the ballot into a sealed ballot box. If the AV-OS detects overvotes (voting for more candidates than the allowed number of candidates in a contest), it can return the ballot to the voter for correction.

  Officials or poll workers configure the AV-OS for each election by inserting a memory card into a slot on the front of the machine. The memory card stores the names of races and candidates, interpreted code used for printing reports, and the running tallies of votes for each candidate. At the end of the election, poll workers remove the memory card from the AV-OS and officials at election headquarters upload the results to a tabulation system to determine the result.

  The AV-OS memory card uses a non-standard interface format but acts only as a passive storage device. It contains all the election-specific information and can be used in any AV-OS machine.

  The AV-OS runs custom election software written by Diebold. The software is a monolithic application that executes directly in a single-threaded fashion on the microprocessor. There is no operating system and no support for multi-user operation, timesharing, or memory protection.

- The *AccuVote-TSX* (AV-TSX) is a DRE voting machine. It interacts with the voter via a touchscreen LCD display, and it supports audio ballots for increased accessibility.

  The AV-TSX is configured for each election by inserting a memory card into a slot behind a locked door on the side of the machine. The memory card is a standard PCMCIA flash storage card. Before the election, the file system on the memory card stores the election definition, sound files, translations for other languages, interpreted code that is used to print reports, and other configuration information.

  As each ballot is cast, the AV-TSX stores an electronic record of the votes associated with that ballot onto a file on the memory card. At the close of polls, the AV-TSX counts all of the votes

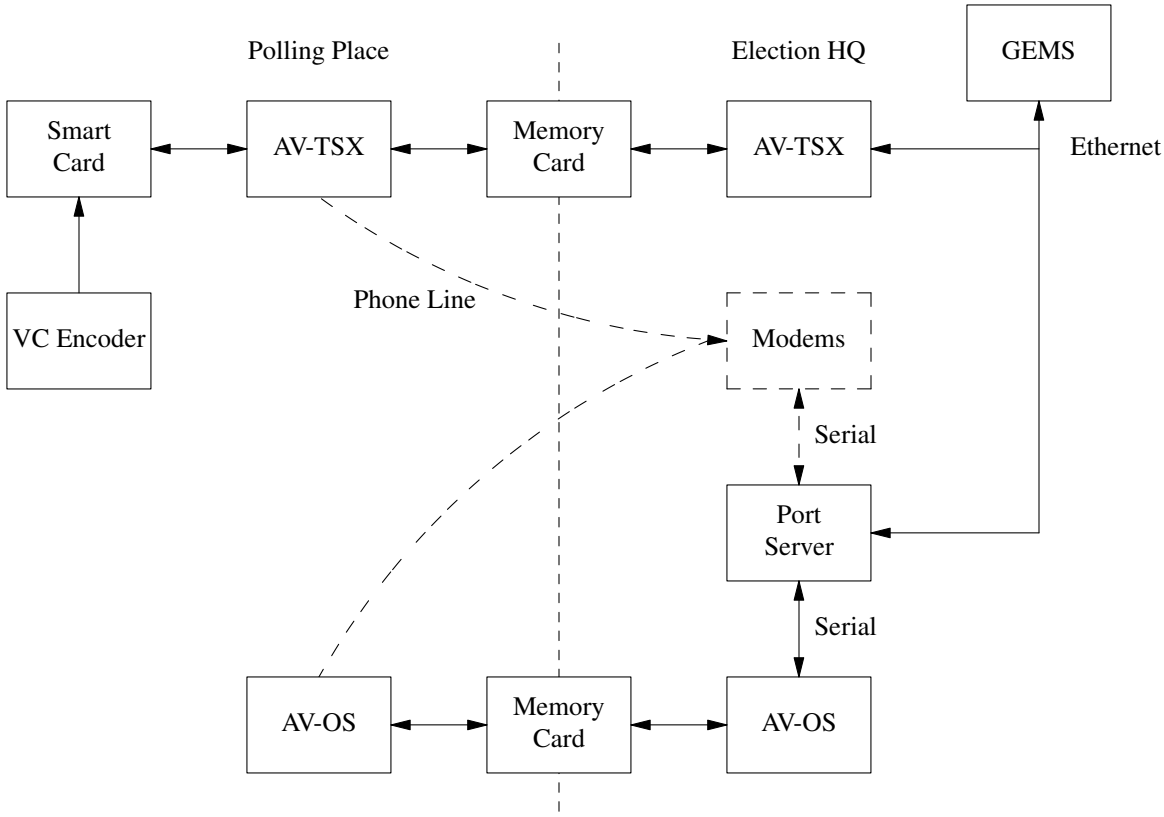


Figure 2.1: **Components of the Diebold system in a typical county.** At election headquarters, there is a GEMS server connected via Ethernet to one or more central-office AV-TSX machines and through a port server to one or more central-office AV-OS machines. These machines read and write memory cards, which are used to transfer ballots to machines at the polling place and to read back election results. Polling places also contain voter card encoders, which program smart cards that allow voters to access the AV-TSX machines. Optionally, modems are used to transfer unofficial ballot results from the polling places.

and prints a summary tape showing the vote tallies. After the election, poll workers remove the memory card from the machine and send it to election headquarters so that the electronic vote records can be uploaded for tabulation.

The AV-TSX also contains a printer attachment that is used for printing a voter-verifiable paper audit trail (VVPAT) corresponding to each ballot cast by the voter. Before casting their ballots, voters have an opportunity to examine printed VVPAT records and confirm that they accurately represent their intent.

Internally, the TSX contains much of the same hardware found in a general-purpose PC. It contains a 32-bit Intel xScale processor, 32 MB of internal flash memory, and 64 MB of RAM. The TSX runs version 4.1 of Microsoft's Windows CE operating system with modifications written by Diebold. An application called *BallotStation* runs on top of the operating system and provides the user interface that voters and poll workers see. BallotStation interacts with the voter, accepts and records votes, counts the votes, and performs all other election-related processing. The TSX also contains a custom bootloader and other low-level support software.

- *Smart cards* are used with the AV-TSX to authenticate voters and poll workers. Each smart card is a piece of plastic in the shape of a credit card with an embedded computer chip that

can communicate with the AV-TSX when inserted into a slot on the side of the machine. Smart cards are used for several purposes:

  - *Voter cards* are used to authenticate voters. When a voter signs in, a poll worker gives them an activated voter card. The voter inserts the card into an AV-TSX, and the machine allows her to cast one ballot. Once the vote has been recorded, the AV-TSX deactivates the voter card so that it cannot be used to vote a second time. The voter returns the card to poll workers, who can reactivate it for subsequent voters.

    Alternatively, in some jurisdictions poll workers activate the voter card and then insert it into the AV-TSX unit for the voter, so that voters do not have to insert it themselves.
  - *Supervisor cards* are used to authenticate poll workers. The chief poll worker would normally be given a supervisor card. When the supervisor card is inserted into an AV-TSX unit, the poll worker is presented with extra functionality not available to voters, such as the ability to close the polls or examine audit logs. Supervisor cards would normally not be provided to voters.

- The *vote card encoder* is a calculator-sized device used by poll workers to generate new voter cards. Poll workers insert voter cards into the device to activate them. Optionally, workers can indicate which ballot style the voter should receive for split precincts or primary elections.

- *Memory cards* are used by the AV-OS and AV-TSX, as described above.

  Practices regarding use of memory cards vary from county to county. Typically, voting equipment might be delivered to the polling place with memory cards already sealed into them. In other words, before the election, county staff program the memory cards with election definition files, insert them into AV-OS or AV-TSX units at the warehouse, and place a tamper-evident seal over the memory card door. Then they ship the AV-OS or AV-TSX units to polling places. Alternatively, memory cards can be provided to poll workers separately and poll workers can insert the memory cards into the AV-OS or AV-TSX units on election morning before the polls are opened.

  After the close of polls, there are several options for how memory cards can be returned to the county. One option is that poll workers can break the seals on the memory card doors, remove the memory cards, and transport them back to county headquarters. Another option is that the equipment can be returned with the memory cards still sealed inside them, and after receiving the equipment at the warehouse, county staff can break the seals and remove the memory cards. Counties normally choose one of these two options.

We note that some counties may not use all of these components. Some counties do not use the AV-OS; in those counties, all (or most) voters vote on AV-TSX machines, and polling places are typically equipped with enough AV-TSX units to handle the expected turnout. Other counties use both the AV-OS and the AV-TSX, and a polling place might contain both an AV-OS for scanning paper ballots and one or more AV-TSX units for accessibility. Counties in the latter category can offer voters the option of voting by paper ballot or by touchscreen, or they can require most voters to use the paper ballot and reserve the AV-TSX unit for voters who need the accessibility or language support it provides.

2.2 Components at Election Headquarters

In addition, there are a number of components present at the county's elections headquarters:

- *GEMS* is an election management software application that runs on an ordinary desktop PC. GEMS is used to control many aspects of the election, including designing ballots, downloading election definition files to voting machines, compiling election results, and reporting the election outcome.

GEMS is a Windows application. Typically, it runs on a PC configured by the vendor running Windows 2000 or Windows XP as well as a number of commodity software applications (e. g., Adobe Acrobat reader). GEMS uses Microsoft's Jet database technology (the database engine used by Microsoft Access).

- The *AccuVote-OS Central Count* is an optical scan machine used for scanning and counting paper ballots at the election headquarters. It is commonly used to scan absentee (vote-by-mail) ballots as well as provisional, damaged, duplicated, or enhanced ballots. The AV-OS Central Count machine connects to GEMS via a serial link, and its operation is controlled by GEMS. It scans ballots and interprets ballot marks, but it then immediately uploads a record of each vote to GEMS and does not attempt to tabulate or keep any record of votes.

  The AV-OS Central Count uses essentially the same hardware as the AV-OS Precinct Count, but the two models run very different software.

  The AV-OS Central Count is normally used in conjunction with the AccuFeed unit, which feeds paper ballots into the scanner at a controlled rate. A small infrared (IR) sensor attached to the AV-OS unit, and then the AccuFeed and AV-OS communicate by IR to control ballot feeding.

- An *Ethernet network* would typically be used to connect many of the devices at the central office. As mentioned below, devices on the Ethernet network might include the GEMS PC, a port server device, AV-TSX units, other PCs (e. g., a PC running JResultsClient, for displaying unofficial election results to observers on election night), and potentially PCs used for unrelated purposes.

  Diebold employees normally set up an Ethernet network and configure the devices on the network for the county. Of course, over the lifetime of the voting system, installation and configuration decisions are at the county's discretion. In some counties this Ethernet network might be strictly isolated. However, we presume it is also possible that this network might be connected to the election department's internal network or the county intranet; we were not provided any detailed information on individual county practices, and we were not able to rule out such a possibility. We did not find any clear prohibition in the system documentation that forbids connecting other devices or networks to this Ethernet network.

- One or more AV-TSX units would normally be connected to the GEMS PC by Ethernet. These AV-TSX units are identical to AV-TSX units used in the polling place, but they serve a different function: they are used to read and write AV-TSX memory cards. We will call any AV-TSX unit that is used in this fashion a "central-office AV-TSX" to distinguish it from an AV-TSX unit that is used by voters. These AV-TSX units in principle can also be connected by serial cable, but Diebold has told us that counties would normally use an Ethernet network instead of a serial cable. County staff insert a PCMCIA Ethernet card into one of the PCMCIA slots on the AV-TSX and then connect the AV-TSX to an Ethernet hub. The BallotStation application provides the result upload capabilities and interfaces with the GEMS server over the network.

  Before the election, once election administrators have laid out the election on the GEMS server, county staff use GEMS and the central-office AV-TSX units to write election definition files onto memory cards. Staff must prepare one memory card per AV-TSX that will be deployed in the field. For instance, a county might have 2000 AV-TSX units that will be deployed in polling sites throughout the county on election day, and might have 5 central-office AV-TSX units used for writing memory cards. County staff would then write those 2000 memory cards by inserting each memory card into a central-office AV-TSX unit, one at a time, and instructing GEMS to make the proper election definition files available for the machine to download. Once all memory cards have been programmed, they can be inserted into AV-TSX units destined for the field.

  After the election, as poll workers return memory cards to county headquarters, county staff use the central-office AV-TSX units to read results files from the memory cards and upload

them to GEMS for tabulation. GEMS inserts the vote data into its database and tabulates the votes.

- One or more AV-OS Precinct Count units would also normally be connected to GEMS by serial cable. These central-office AV-OS units serve a purpose analogous to that of the central-office AV-TSX units. They are controlled by GEMS and used to read and write memory cards intended for use with AV-OS units in polling places.

  Note that AV-OS memory cards are not compatible with AV-TSX memory cards, since they are of a different shape and use a different technology. Therefore, AV-OS memory cards must be read and written by central-office AV-OS units, while AV-TSX memory cards must be read and written by central-office AV-TSX units.

- One might also find a device used to expand the number of serial links that can be connected to the GEMS PC. The sample GEMS setup examined by the Red Team used a *Digi PortServer II*, which is an embedded device that connects to the GEMS PC via an Ethernet network (using TCP/IP) and provides up to 64 serial ports that can be connected to central-office AV-OS units or modems [3]. Alternatively, one might find a Digi device that works similarly except that it is connected to GEMS by a serial link instead of via an Ethernet network. These are commodity devices sold on the open market. They are used to expand the number of AV-OS units that can be connected to GEMS. Because ordinary PCs normally have only one serial port (or at most a few serial ports), this provides a way to connect many AV-OS units to the single GEMS PC.

- Several types of *smart cards* are also used at county headquarters to authenticate county staff to the AV-TSX units. In particular:

  - *Central election administrator cards* are used to authenticate county staff. Insertion of a central administrator card into a AV-TSX unit yields access to additional functionality, such as the ability to configure the AV-TSX unit. These cards would normally not be provided to voters or poll workers and would be closely held by county workers.
  - *Security key cards* are used to update the cryptographic keys on the AV-TSX. These cards are security-sensitive and should only be available to trusted county staff. Security key cards are created using the *Key Card Tool*, a Windows software application installed on a PC in county headquarters.

  Note that, while there are four types of smart cards in total, their internal components are physically identical: the four types of smart cards differ only in the data that has been written to them, and in the label that is printed on their exterior. When a smart card is inserted into the AV-TSX, the AV-TSX uses the data it reads on the card to determine what type of smart card was inserted.

- One might find *modems* that can be used to accept communications over the public telephone network. We discuss modems in detail in Section 4.1.8.

CHAPTER 3

Major Attacks

In the course of our source code review, we identified numerous issues that might allow an attacker to compromise the integrity, reliability, and secrecy of elections run on Diebold systems. These problems are compounded since attackers can exploit multiple vulnerabilities in combination to carry out more powerful attacks. In this chapter, we discuss two kinds of combination attacks that we believe are among the most serious that we have identified. The first attack could allow a single technically sophisticated person with limited access to election equipment to spread a voting machine virus to all machines within a county. A virus could subtly switch votes from one candidate to another, or cause widespread disenfranchisement by overwriting the machines' firmware. The second attack could enable election officials or other insiders to violate ballot secrecy and discover how voters voted.

3.1 Voting Machine Viruses

Like desktop PCs, computer voting machines are vulnerable to viruses. Viruses are malicious software programs that spread automatically from machine to machine. Viruses pose a particularly severe threat to voting security because they can spread invisibly in the background, even when procedural safeguards that limit physical access to the machines are followed. We believe it would be possible for a sophisticated attacker, by exploiting several of the vulnerabilities that we discovered, to launch a powerful virus that would spread from even a single infected voting machine to all the AV-OS and AV-TSX machines within a county.

In prior work, Feldman, et al. [14] demonstrated a working voting machine virus that could spread automatically between AccuVote-TS units if the machines were booted with infected memory cards inserted. From our review of the source code, it appears plausible that the attacks Feldman, et al. identified remain feasible on the AccuVote-TSX. Other vulnerabilities that we identify in this report create further avenues for spreading viruses that are potentially more dangerous than previously known mechanisms.

Creating a voting machine virus like the one we describe below would require moderate to sophisticated programming skills and access to voting equipment, but both are likely available on the black market. A single person with these capabilities could create a virus. An AccuVote-TS was recently listed on eBay, and an attacker unable to purchase one could attempt to steal one instead. Machines purchased or stolen from other states could be just as useful to attackers as ones from California (if the machines had the same software version), so improving physical safeguards within the state will only have limited benefit.

We now describe one scenario where a voting machine virus could spread throughout a county's election system (see Figure 3.1). Many variations on this scenario are possible, so attempts to fix this problem must not focus exclusively on the specifics of this attack.

1. **Initial infection of an AV-TSX**

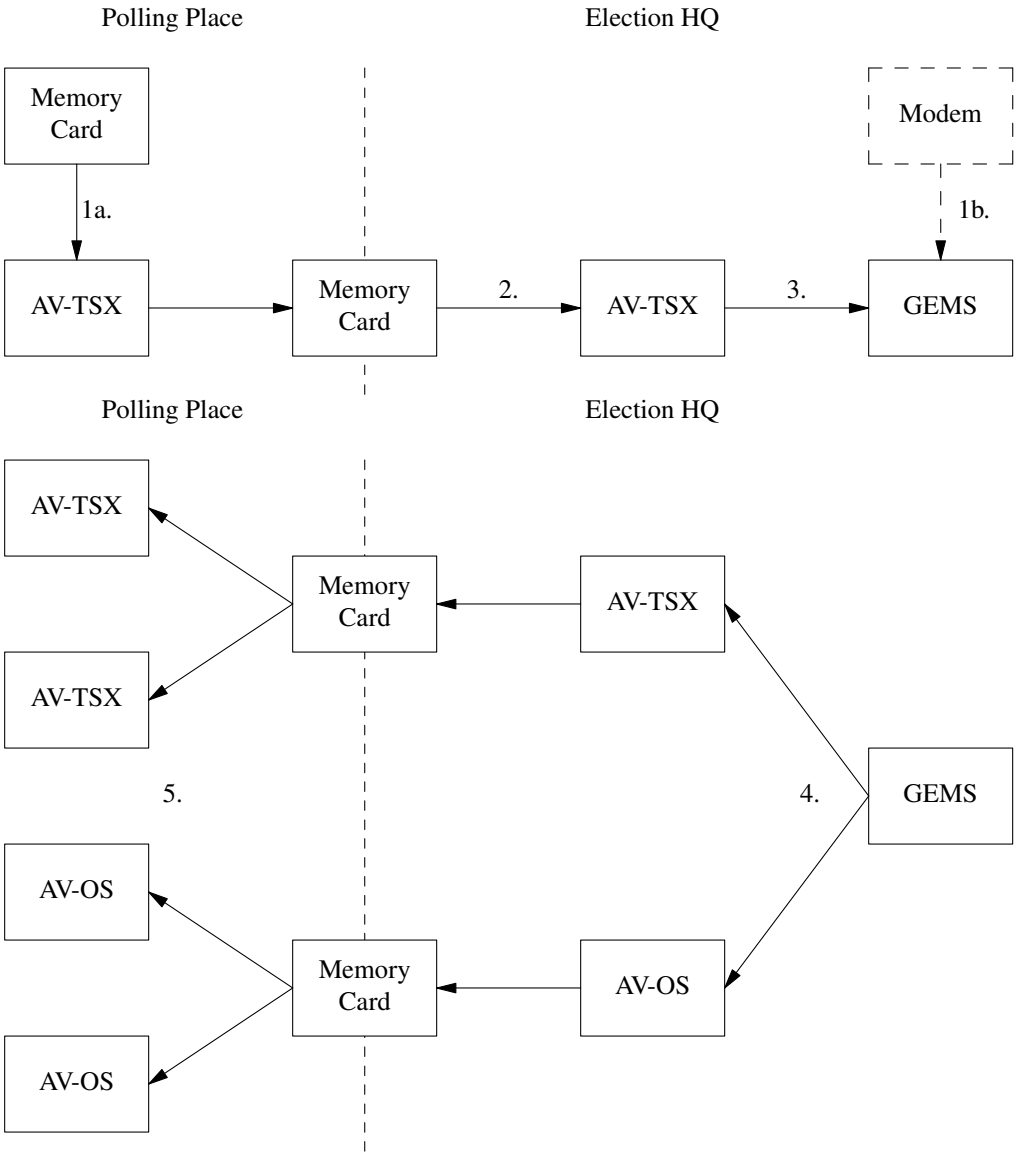


Figure 3.1: **Propagation of a virus over the course of two election cycles.** During the first election: (1a) An attacker temporarily inserts a memory card containing a voting machine virus into an AV-TSX, infecting the machine. (2) After the election, poll workers remove the memory card containing ballot results from the infected machine and send it to election headquarters for tabulation; the virus has corrupted the files on the card, so inserting it into a central-office TSX infects that machine. (3) The infected central-office TSX attacks the GEMS PC over the Ethernet network by using known vulnerabilities in Windows; when the attack succeeds, the virus infects the GEMS server. (1b) Alternatively, if the county uses modems to return unofficial election results, an attacker can target the GEMS server directly over the modem connection, infecting it directly.

During the next election cycle: (4) The virus running on the GEMS server infects memory cards when officials download the new ballots; these cards are placed in voting machines throughout the county. (5) On election day, the virus executes its payload, which may involve altering votes or otherwise disrupting the election.

The attacker, after developing the virus in advance of the election, needs only momentary physical access to an AV-TSX or memory card in order to initiate the infection.

One way to infect the initial machine would be to exploit the insecure software update mechanism described in Issue 5.2.1 or Issue 5.2.2.[1] If memory cards ship separately from machines, the attacker could intercept a memory card en route and copy the virus onto the card. The virus would be installed on the AV-TSX when the machine boots with the card in place. AV-TSX memory cards are commodity PCMCIA cards that can be bought on the open market and read and written using any laptop, so no special equipment is required to mount this attack. If, instead, machines ship with memory cards sealed in place, the attacker would need to gain physical access to a machine, break the seal and unlock the lock, replace the memory card with one containing the virus, reboot the machine to install the virus, reinsert the original memory card, and relock the enclosure. Though this may sound complicated, the Red Team has told us that software updates can be installed with less than one minute of physical access in a manner that would likely raise minimal suspicion from poll workers. The only physical evidence would be the single broken seal.

Some counties use modems to transmit unofficial election results back to election headquarters after polls close. In this case, an attacker could attempt to infect the GEMS server directly by connecting to it over the modem. As discussed in Section 4.1.8, this would allow the attacker to skip directly to step 4 below.

2. **Viral spread to the central-office AV-TSX**

   After the election, officials remove the memory cards from each machine and take them to the election headquarters, where a small number of AV-TSX machines are networked to a GEMS server. The next step in the virus's lifecycle is to infect these central-office AV-TSX units.

   On the initially infected AV-TSX, the virus can manipulate the election database file stored on the removable memory card by exploiting Issue 5.2.6. The attacker could design the virus to corrupt the file to exploit problems, such as Issue 5.2.16, that allow the execution of arbitrary code during the result upload stage. Later, officials take the memory card with the manipulated election description and place it into a central-office AV-TSX. When officials initiate the upload function, the attacker's code executes and infects the central AV-TSX machine with the virus. A well-crafted virus might be able to do this without causing any visible signs of foul play.

3. **Attacking the GEMS machine**

   As soon as the virus infects the central-office AV-TSX, it can begin attacking the GEMS machine. In a typical deployment, as described by Diebold, the GEMS machine and the central-office AV-TSX machines attach to a single Ethernet switch and communicate using TCP/IP. This means that the GEMS PC exposes a large attack surface to the AV-TSX. Vulnerabilities in the PC's operating system (Windows), network drivers, and network services could all be attacked. The hacker community is already aware of exploitable flaws in some of these components. Even if automatic patches exist for these commodity components, the PC's software may not be up-to-date.

   The Red Team's report describes how they were able to use widely available exploit tools to exploit holes in Windows and take control of the GEMS PC from another PC on the same subnet [3]. A virus running on the central AV-TSX could be programmed to perform a similar attack. After gaining control of the GEMS PC, the virus would install itself and proceed to the next phase of its lifecycle. It could hide itself from system administrators and from common security tools[2] using rootkit techniques [16].

[1] For example, the initial infection might replace the machine's bootloader software. This bootloader could then install high-level infection software [14].

[2] Standard anti-virus software would be unlikely to detect a special-purpose voting machine virus that had infected the GEMS server. Such software does not exist for the AV-TSX and AV-OS.

4. **Spreading back to the field**

   At the beginning of the next election cycle, the infected GEMS system can spread the virus to the voting machines used in the field. It might spread to AV-TSX systems by tampering with the election data files as they are downloaded to memory cards that will be distributed to polling places. By introducing deliberate errors into these files, the virus could exploit vulnerabilities (e. g., Issue 5.2.13, Issue 5.2.15) that will allow virus code to execute on the systems during voting. The virus could also spread to AV-OS Precinct Count machines in a similar manner by exploiting Issue 5.1.2. Since typical procedures call for every memory card used in the county to be created using the GEMS server, this step would allow the virus to infect every AV-OS and AV-TSX machine used by voters.

3.2 Virus Payloads

What harm can a voting machine virus or other wide-scale compromises do? Among the most dangerous payloads would be an attempt to shift a close race by subtly stealing votes and an attempt to disrupt an election by launching a large scale denial-of-service attack. Procedural countermeasures might not be sufficient to defend against these attacks, as discussed in Chapter 6.

1. **Subtle vote stealing**

   An attacker could use a voting machine virus to reprogram a large number of AV-TSX or AV-OS machines to steal votes. When programming the attack, the attacker could decide which votes to steal (e. g., from particular candidates, races, or parties), how to steal them (e. g., by adding, deleting, or switching votes from one candidate to another), and when to execute the attack (e. g., only in closely contested races, or only in precincts with certain voting patterns). California's mandatory voter-verifiable paper audit trail (VVPAT) provides a valuable defense against electronic vote stealing, but it will not necessarily be able to detect and correct every kind of attack — particularly in races with a narrow margin of victory.

   In a close election, one particularly dangerous scenario would be a widely-spread virus that subtly shifts votes between candidates on both the paper and electronic records. Suppose the candidates are named Alice and Bob. A Bob supporter could reprogram the machines to look for voters who select Alice. One percent of the time, after the voter has selected Alice, they machines could behave as if the voter had picked Bob, displaying a vote for Bob on the confirmation screen and on the printed paper record. A cleverly designed virus wouldn't interfere with an attempt to correct the problem, so voters who notice the error could cancel the printed VVPAT record and change the selection back to Alice.

   An attack like this might shift enough votes to cause the wrong result in a close election, but could it really be done without being detected? Several factors favor the attacker. First, assuming that only a small fraction of voters would carefully review the paper VVPAT record, many voters would overlook the problem and allow incorrect votes to be recorded in both the electronic and paper records. Second, while a few voters might report the problem to poll workers, election officials would have difficulty determining whether the cause was voter error or a problem with the machines. This is similar to problems that Sarasota County, Florida experienced with its DRE voting machines in the November 2006 election [11]. In one race during that election, hundreds of voters reported that the machines displayed the wrong selection on the summary screen, or that they failed to show the race on the summary screen at all. Some observers eventually concluded that the cause was voter error due to a poorly designed ballot layout.

   Even if officials suspect an electronic attack, the virus author could take countermeasures to thwart later investigation. The attacker could tamper with the system logs to remove traces of the virus's activity, and remove the virus after the election when the machine powers on again. By the time an investigation is commenced, most of the evidence of the problem could be destroyed.

Finally, even in the best case when officials do detect the virus, they might have difficulty undoing its effects without holding a new election — thus, the vote stealing attack becomes, at best, a massive denial of service attack. It would probably be impossible to tell how many votes had been shifted as a result of the attack, since the electronic and paper records would both reflect the fraudulent result.

A virus could also shift vote totals in the reports produced by AccuVote-OS Precinct Count machines. Conceivably, this would need to be paired with a corresponding attack that alters the paper ballots. This attack is considerably more difficult to accomplish on a large scale since the AV-OS scanner is unable to alter ballots.

Another notable but less-damaging attack by a virus author would be to re-program the AV-OS to selectively allow overvotes for a disfavored candidate. A voter who accidentally overvotes would not be notified by the AV-OS of her mistake and would not be given a chance to fix her ballot. After the election, if an overvoted ballot is rescanned, the overvote would likely be deemed invalid by officials. This attack might raise suspicion since AV-OS-scanned ballots should have been previously checked for overvotes, but the damage would be done by the time the attack was detected.

Other means of attacking the VVPAT are discussed in Section 3.3.

2. **Massive denial of service**

   Rather than stealing votes directly, attackers might choose a more passive strategy and attempt to disrupt the election process itself by disabling machines, destroying vote records, or slowing down voting. These attacks could be targeted at precincts that are likely to support an opposing candidate (or even triggered only after the virus detects that the opposing candidate has won a certain portion of the votes on a machine). Alternatively, the attack could be carried out indiscriminately in hopes of causing such widespread disruption that the election would be postponed.

   What kinds of disruption could a virus cause? The AccuVote-TSX memory architecture exposes important system code, including the bootloader, operating system kernel, and voting software, to tampering by other software running on the machine. Malicious software on the machine could overwrite the bootloader or other parts of the software, rendering the machine inoperable. Repairing this damage would require a visit from a service technician and possibly even a return to the factory. Feldman, et al. demonstrated how such a denial of service attack was possible with the AccuVote-TS [14].

   An attacker would have many choices about when and where to trigger an attack and what kind of damage to do. Some attacks might be very difficult to distinguish from non-malicious hardware and software malfunctions.

   Denial-of-service attacks on the AccuVote-OS machines would be slightly less damaging. If an attack causes the AV-OS machines to break down, only voters who invalidly fill out their paper ballot (e. g., overvote) will be affected, since they will not be warned or given the chance to correct their mistakes.

3.3 Attacking the VVPAT

The design of the VVPAT mechanism used by the AV-TSX poses a threat to the secrecy of voter ballots and places a limit on its ability to detect malicious software. Two aspects of the design are especially problematic:

- The AV-TSX contains a movable flap over the VVPAT window and can be open or closed. When it is closed, the voter cannot see the VVPAT record and is unlikely to know in advance that the flap needs to be opened. If the flap is closed when the voter walks up to the machine, they may cast their ballot without checking the VVPAT record and without even being aware

that they could have checked the VVPAT record. The flap closes easily, and once it closes, it might stay closed for many voters.

Consequently, the presence of this flap may reduce the number of voters who check the VVPAT record. This makes audits less effective because it undermines the presumption that the VVPAT record accurately represents the voter's intent. This flap may also heighten the damage done by paper jams and printer failures: if the flap is closed when the printer jams, then several voters may continue voting before anyone notices that multiple VVPAT records have been destroyed and rendered unreadable by the unnoticed printer jam.

- The AV-TSX VVPAT uses a reel-to-reel printer mechanism. The system contains a spool of blank thermal paper which feeds through the printer mechanism, then past the window where it is visible to the voter, and then winds onto a second take-up spool. The machine records votes continuously on the spool of paper without cutting it in between voters. Consequently, election workers have an opportunity to associate voters with the paper records by matching the order in which voters used the machine to the order of the records on the paper.

Most voters only vote at a polling place once or twice every two years. Consequently, each time they use the equipment, they will effectively be learning for the first time how to use it. Voters cannot be expected to know in advance the intricacies of how the machines work: instead, they will likely be learning this as they go along. This creates the possibility that malicious software on an AV-TSX could trick voters by subtly deviating from the normal protocol for printing VVPAT records.

To illustrate the risks, we list four hypothetical attacks that an attacker who has subverted the software on the AV-TSX could mount. This list of example attacks is not intended to be exhaustive or comprehensive.

1. In our first attack, the malicious AV-TSX behaves honestly for 90% of the voters, and randomly selects 10% of the voters to cheat. For those unlucky voters, it behaves legitimately except that it always prints candidate X on the VVPAT record, whether the voter selected candidate X or his rival. If the voter notices the error and spoils her ballot, the machine allows her to go back and change her selection, and it behaves honestly for the remainder of that voter's session. If the voter does not notice, the machine casts the ballot. The machine records an electronic vote record matching whatever is printed on the VVPAT record.

   If we assume that perhaps 50% of voters will not bother to check what is on the VVPAT record, the machine will succeed in stealing up to 5% of the votes, which may be enough to overturn the outcome of a close race if every machine in the county contains this malicious software. Moreover, this attack leaves no permanent evidence that would be detected during the 1% manual tally or the official canvass. In principle, an election official who counted the number of spoiled VVPAT records might notice the increase in spoiled VVPAT records [4], but we are not aware of any county that currently performs this check as part of the official canvass.

   We are not convinced that increasing the number of spoiled VVPAT records by an amount equal to 5% of the total number of ballots would be enough to provide a clear indication of fraud. Even if suspicions were raised, the evidence might still admit multiple possible interpretations and thus, there might be no indisputable evidence of fraud that a court could use to throw out the election results. Furthermore, even if the attack is detected and a court is persuaded to order a new election, the controversy could undermine voter confidence. This is an example of an attack that cannot be mounted against manually-marked paper ballots.

2. In our second attack, a malicious AV-TSX unit cheats 5% of the voters by deviating from the normal desired operation in only one small way. When the unlucky voter casts her ballot, the machine does not scroll the VVPAT record up into the security canister. It simply leaves the VVPAT record showing in the glass window and prints a message on the screen saying "Your vote was recorded. Thank you for voting." Shortly after the voter walks away, the

machine prints "CANCELLED" under the VVPAT record to spoil it, scrolls it up into the security canister, prints a new VVPAT record containing votes for the attacker's preferred candidates, and scrolls that into the security canister. This might be difficult to detect.

3. Alternatively, one can imagine that when the unlucky voter casts her ballot, the AV-TSX machine immediately prints "CANCELLED," then prints a new VVPAT record, and scrolls that up into the security canister, all in one action. If the printer scrolls the paper fast enough, it might be difficult for a first-time voter to notice what went wrong. If an occasional unlucky voter does happen to notice the misbehavior, she cannot demonstrate to anyone else that the machine cheated her: by the time she can call over a poll worker or another witness, the evidence is gone and it is too late. Even if a poll worker peers over the shoulder of the next few voters (violating their ballot secrecy) to see if the problem recurs, it is unlikely that the poll worker will notice any further problem due to the small number of voters targeted. We fear that even if a few voters do notice the issue and complain, their complaints might be discounted or there might be little that poll workers can do about it.

4. Yet another possible attack applies to counties that use DREs for provisional voting. Suppose that the attacker wants to favor candidate X over candidate Y and is able to introduce malicious software onto the county's AV-TSX units. When a provisional voter steps up to use the machine, the malicious software observes which candidate the voter selects. If the voter selects candidate Y, the machine behaves honestly and prints a VVPAT record that is correctly marked as provisional, prints the challenge code associated with this provisional voter, and correctly records the vote electronically as a provisional vote. However, if the provisional voter selects candidate X, then the machine prints a VVPAT record and makes an electronic record as though this were a non-provisional voter. Later, when a subsequent non-provisional voter tries to vote for candidate Y, the machine prints a VVPAT record and makes an electronic record as though that subsequent voter were a provisional voter (using the challenge code associated with the earlier provisional voter). In effect, each provisional voter who votes for candidate X is matched up with a non-provisional voter who votes for candidate Y, and the provisional status is swapped. Later, during the resolution of provisional voters, if some provisional votes are discarded because the voter was not eligible to vote, then officials will be discarding votes for candidate Y that should have counted and retaining votes for candidate X that should have been discarded. The number of votes that can be stolen in this way is governed by the number of provisional voters whose votes are ultimately rejected. If this number exceeds the margin of victory, this attack strategy can change the outcome of the election. This kind of attack would not be detected by the 1% manual tally or during the official canvass because the electronic records do match the VVPAT records exactly and because the number of voters and provisional voters are not affected by the attack. The only chance to detect this attack is for voters to notice the provisional status of their votes, but it is not clear whether voters would notice this or would understand the consequences if they did.

Malicious software might also try to increase the odds of avoiding detection by targeting its attacks towards voters who are less likely to notice the fraud. For instance, instead of picking voters to cheat at random, the malicious software might watch for voters who appear to be having trouble (as evidenced, for instance, by their use of the "Help" button, by slow progress through the ballot, or by multiple attempts to change their selections) and selectively defraud these voters.

Would these attacks succeed in avoiding detection? We do not know. To the best of our knowledge, it is an open question whether voters would notice. We are not aware of any studies that have tested these attacks, but there are reasons to be concerned [13]. It is possible that none of these attack strategies would succeed. We would like to believe that these kinds of attacks would be detected. However, it is also possible that an attacker who studied human behavior could come up with methods to steal elections without detection. The point is that we do not know whether malicious software could fool voters into accepting fraudulent VVPAT records. Given that the security of the AV-TSX relies upon the assumption that the VVPAT record will accurately represent

voter intent, this uncertainty surrounding VVPATs and voter behavior may be of concern. It seems to cast some doubt on how much we can rely upon the VVPATs.

Of course, these kinds of attacks are only possible if the attacker can find a way to replace the software on the AV-TSX. However, the VVPAT was intended as an independent check upon the operation of the machines, and these risks undermine the independence of the VVPAT records. These attacks are possible because the VVPAT printer and spool are entirely under software control, so if the software is subverted, it can control how the VVPAT record is printed. This architectural feature of the AV-TSX is unfortunate. It might be better to have an architecture that left no plausible avenue for an attacker to subvert the paper trail.

Manually-marked paper ballots scanned with the AV-OS are not subject to these kinds of human factors risks. Because the process of marking the ballot does not involve any interaction with complex technology, there is no opportunity for corrupted devices to try to influence the voter ballot marks. The marks on the paper ballot record the voter's intent unmediated by technology. This may make manual recounts and the 1% manual tally of the AV-OS more effective and less susceptible to subversion by malicious software. Consequently, it seems plausible that voting systems based upon the AV-OS may prove to be more resilient to technical attack than voting systems based upon the AV-TSX.

3.4 Attacking Ballot Secrecy

In addition to threats to the accuracy of election results, we are charged with identifying problems that could threaten the secrecy of voter selections. Secrecy makes it difficult for voters to sell their votes, since they can't prove to anybody else how they voted. Ballot secrecy also helps voters stand up to intimidation by those who threaten to harm them if they do not vote a certain way. We found a variety of issues with the AV-TSX that pose significant threats in this area.

As we describe in detail in later sections, the machine stores votes in the order in which they were cast (Issue 5.2.19); it stores them together with a record of the time they were cast and, if a specific configuration option is enabled, prints this time in a barcode on the paper VVPAT record (Issue 5.2.20); and it assigns them each an encrypted serial number that can be decrypted to discover the order of voting (Issue 5.2.21). Any one of these problems could leak enough information about the votes to reveal how individuals voted.

Exploiting these problems would require three resources. First, an attacker would need access to the voting data — either the barcoded VVPAT records or the election results file from the memory card or voting machine. Second, if attacking the election results file, the attacker would need to know the data key used to encrypt the results file and generate ballot serial numbers; Issue 5.2.5 explains how an attacker with access to a single voting machine can determine this county-wide key. Third, an attacker would need to know on which machines target individuals cast their votes, as well as the time of their votes or their positions in the sequence of votes cast. For example, in a targeted attack, a human observer or hidden camera could observe how many people voted on a machine before the targeted individual. In a broader breach of privacy, the attacker could learn the order of voters from the polling place sign-in list, if that list records the order in which voters sign in. There are several ways that an attacker could obtain this information, but we are particularly concerned that all of the necessary items could be obtained with relative ease by corrupt poll workers or election officials.

Of course, most poll workers and election officials are honest. Poll workers volunteer their time for what is a critical but largely thankless job. What concerns us is that a malicious person who wants to attack the election can purposely volunteer as a poll worker in order to obtain access to sensitive data. Regardless of whether or not any poll workers actually are malicious, the fact that the AV-TSX makes it possible for malicious officials to determine how individuals voted may be detrimental to voter confidence and participation.

CHAPTER 4

Systemic and Architectural Issues

Given the consequences of election fraud and the importance of public confidence in elections, voting systems and software must be designed from the ground up to be secure. Building a secure system involves identifying the threats that it could face and producing a design that not only counters those threats but employs defense-in-depth to limit the damage that any undiscovered vulnerabilities could cause. It also requires the use of defensive programming techniques to minimize software defects and the use of sound software engineering practices to ensure that software developers are properly trained and that source code is properly reviewed before release.

In our analysis of the Diebold system, we found significant systemic weaknesses in its design and implementation as well as in the engineering practices used to develop it. Our analysis is based both on our direct examination of the system's source code and on an interview that we conducted with Talbot Iredale, Software Development Manager, Diebold Election Systems [1].[1]

4.1 Design

4.1.1 Large Attack Surface

Experienced security practitioners often also recommend analysis of the "attack surface" of a software system. The attack surface is the interface that is exposed to the attacker. This includes all operations that the attacker can invoke, any data that the attacker can control, protocols that the attacker can participate in, and so forth. The larger the attack surface, the more degrees of freedom the attacker has in crafting attack strategies. A bug in any code that is exposed to an attacker may lead to an exploitable vulnerability, so a large attack surface also means that a large volume of code is security-critical. Consequently, systems with a large attack surface tend to be more prone to security vulnerabilities.

The Diebold voting system has a large attack surface. Exposed interfaces include:

1. The user interface on the AV-TSX.
2. The protocol spoken between the AV-TSX and the smart card.
3. The content of election database and other files on the AV-TSX memory card, as read by AV-TSX units in the field.
4. The content of the ballot results files on the AV-TSX memory card, as read by other AV-TSX units.
5. The data transmitted between GEMS and a central-office AV-TSX, when the two are connected by Ethernet or a serial link or modem.
6. The protocol spoken between the smart card and the Voter Card Encoder.

[1] We wish to thank Mr. Iredale for his time and his useful insights.

7. The marks on the paper ballot, as scanned by the AV-OS.[2]
8. The election configuration and other data on the AV-OS memory card, as read by AV-OS units in the field.
9. The election results data on the AV-OS memory card, as read by central-office AV-OS units.
10. The data transmitted between GEMS and a central-office AV-OS, when the two are connected by a serial link or by modem.
11. The interface between multiple GEMS installations during regional processing.

Some of these interfaces are complex and present many opportunities for attack. All of them could potentially be manipulated by an attacker. Given this, one would expect that the risk of exploitable vulnerabilities is high. That expectation was borne out during our examination of the source code.

Our analysis of the risks associated with each of these exposed interfaces is as follows:

1. The user interface on the AV-TSX is complex. It seems to have been implemented carefully for the most part, although we did find one buffer overflow that appeared to be possible to exploit (see Issue 5.2.17).
2. The protocol spoken between the AV-TSX and the smart card is fairly simple. The code was not written defensively (see Section 4.2.2 for a definition of defensive programming) but appears to be free of noticeable security vulnerabilities. However, at the protocol level, the design does not appear to have been as successful (see, e. g., Issue 5.2.7, Issue 5.2.8, and Issue 5.2.9).
3. The content of election database and other files on the AV-TSX memory card, as read by AV-TSX units in the field: The format of these files is complex and rich in features, so this is an especially dangerous area for vulnerabilities. The election database contains a marshaled version of a complex data structure, leaving many opportunities for vulnerabilities in the demarshaling code and in malicious election database files that violate expected invariants. The code that reads these files was not written defensively and we suspect that developers may have failed to consider the possibility that the memory card could contain malicious data. We found many design- and implementation-level defects in the code that reads these files. See, e. g., Issue 5.2.1, Issue 5.2.2, Issue 5.2.3, Issue 5.2.13, Issue 5.2.15, and Issue 5.2.14.
4. The content of the election results files on the AV-TSX memory card, as read by other AV-TSX units: The format of these files is of medium complexity. The code involved in reading these files was not consistently written using defensive programming and we found implementation-level defects with serious consequences in this portion of the file. See, e. g., Issue 5.2.16.
5. The data transmitted between GEMS and a central-office AV-TSX, when the two are connected by Ethernet or a serial link or modem, involves a complex protocol. The code involved in interpreting it appears to validate most inputs, but there were some exceptions: we did find several implementation-level defects that would allow an attacker to mount attacks across this interface. See, e. g., Issue 5.2.18.
6. The protocol spoken between the smart card and the Voter Card Encoder appears to be fairly simple.
7. The marks on the paper ballot, as scanned by the AV-OS, present a fairly simple interface and we did not see much opportunity for malicious manipulation of this data to subvert the security of the AV-OS.

[2] In this section, AV-OS refers to the AV-OS Precinct Count machine, not the AV-OS Central Count machine.

8. The election configuration and other data on the AV-OS memory card, as read by AV-OS units in the field: The format of these files is of medium complexity. The code was not written defensively and we found several design- and implementation-level defects in the code that reads these files. See, e. g., Issue 5.1.3, Issue 5.1.5, Issue 5.1.9, Issue 5.1.10, and Issue 5.1.11.

9. The election results data on the AV-OS memory card, as read by central-office AV-OS units: The format of these files is fairly simple, but the code was not entirely free of vulnerabilities. See Issue 5.1.6.

10. The data transmitted between GEMS and a central-office AV-OS, when the two are connected by a serial link or by modem, involves a communications protocol that is of medium complexity. We did not study this code in any depth.

11. The interface between multiple GEMS installations during regional processing: We did not study this code in any depth.

It is interesting to note that the attack surface of the AV-TSX appears to be larger than that of the AV-OS. This is partially a consequence of the fact that the AV-TSX provides more functionality, but it is also a consequence of the way that users interact with these devices. One might predict, based on this analysis, that the AV-TSX would be at greater risk of attack than the AV-OS.

4.1.2 Complexity

The Diebold system is a complex computing system. Complexity is the enemy of security. All code has bugs; the only way to be sure that software will be secure is to arrange for its design and implementation to be so simple and so small that one can inspect all of it and be confident that all of the bugs and defects in the code are found. By that criterion, the Diebold software is too complex to secure. Put another way: If the Diebold system were secure, it would be the first computing system of this complexity that is fully secure.

One crude measure of software complexity involves counting lines of source code. As may be seen from Table 1.1, the AV-TSX (with 136K SLOC, not counting the COTS OS) is a more complex codebase than the AV-OS (with 20K SLOC, and no OS). This provides a second reason why one might expect the AV-TSX to be at a greater risk of security vulnerabilities than the AV-OS. This was at least partially borne out by our analysis of the source code, as mentioned above.

One principle of secure design is to architect the software so that it has a small Trusted Computing Base (TCB). The TCB is that portion of the software whose correctness suffices to ensure that the system security requirements will be met. The system must be designed to ensure that the TCB cannot be bypassed or subverted. That is, the TCB must be protected from attack and must be written to ensure that the rest of the system cannot violate the security policy even if the rest of the system is compromised or malicious.

The Diebold software not appear to have any clearly defined TCB. It is a monolithic system, with no clear trust boundaries. It does not use compartmentalization. Similarly, we found little evidence of any attempt to apply *defense in depth* or to follow the *principle of least privilege*, two standard principles of secure system engineering.

Due to this architecture, a breach of any part of the software may lead to security violations and breaches of the rest of the software. In this sense, the system is fragile. It is like an oceanliner built without watertight doors: a hole anywhere below the waterline is liable to sink the entire ship. Because code of any significant complexity or scale inevitably has bugs, defects, and flaws, this architecture makes it all but inevitable that the Diebold voting software will have exploitable security vulnerabilities.

4.1.3 Misplaced Trust

In our judgment, the Diebold software places too much trust in people and other components of the system. For instance, the software trusts — relies upon — the memory card to contain files from a legitimate, authorized source. In other words, the software is written with the expectation that

the contents of the memory card come from a benign source, and the software does not effectively defend itself against malicious files on the memory card. That trust seems misplaced: it is too easy for an attacker to tamper with the contents of a memory card. When that expectation is violated, the integrity of the software can be breached.

This theme appears throughout the voting system. In many places where two components communicate, both components rely on each other to be benign, which renders them vulnerable to attack if the security of the component happens to be breached. For instance, the GEMS server trusts the central-office AV-OS and AV-TSX units and everything else that is connected to its own Ethernet network. This trust is dangerous. While those devices might be protected against physical tampering, they must handle data that comes from the field and thus might be malicious. Those devices are at heightened risk of subversion, and it would be safer if GEMS and other system components were written to defend against subversion by malicious devices on the same network.

This risk is especially pronounced if county practices involve taking AV-OS or AV-TSX units that were used in the field in a prior election and connecting them to GEMS in a future election. We cannot realistically protect units in the field from physical tampering. Therefore, we must assume that any unit that has spent time overnight at a polling location could have been tampered with by an attacker and its firmware corrupted. If a compromised AV-TSX unit were later plugged into the GEMS Ethernet network and repurposed as a central-office AV-TSX, at that point a malicious device would be plugged into the GEMS Ethernet network. While GEMS could have been written to protect itself from attack by other devices on the same network, it was not. Once any malicious device is allowed to connect to the central-office Ethernet network, there are no effective technical barriers preventing all other devices on the network from being penetrated.

As another example, all AV-OS and AV-TSX units rely upon — are vulnerable to attack by — the central-office GEMS election management system. GEMS can silently program memory cards so that they contain firmware update files or malicious election database files that will replace the running software on every unit in the field with malicious, corrupted software.[3] This means that the consequences of a breach of GEMS security are more severe than they need to be. For instance, while we might be prepared to entrust the GEMS operator not to manually adjust vote tallies (provided that such actions would be logged and could be detected), there is no reason why a rogue GEMS operator should be granted the power to undetectably replace the software on every AV-OS and AV-TSX unit in the county. Yet this is effectively what the Diebold voting system allows.

This type of pervasive trust makes the Diebold system brittle: a small security breach can have large consequences out of proportion to the initial breach. That, in turn, places an unnecessary burden on procedural protections, because even a brief violation of procedure or a small, seemingly negligible breach of the chain of custody can have disproportionately harmful effects.

4.1.4 Bidirectional Information Flow

The Diebold voting system includes a bidirectional flow of data. Information flows from GEMS to every unit in the field (via memory cards), and from all units in the field back to GEMS (again, via memory cards). This poses some risk of viral spread of infection. In particular, if (1) a memory card or unit in the field can be corrupted and (2) there are any exploitable flaws in the handling of data on the memory card, then a virus may be able to spread from one unit in the field to GEMS and then back to every unit in the field. In practice, we found that both prerequisites are met. Due to the complexity of the data on the memory card, any system of this architecture seems to be at high risk of viral spread. The bidirectional flow of data heightens the impact of these vulnerabilities by allowing viruses to spread throughout the system.

This is not a necessary property of a voting system. For instance, it would be possible to have one central-office application for programming memory cards for distribution to the field and a second application for reading memory cards from the field and tabulating results, with firewalls to ensure that any penetration of the second application cannot affect the first application. See

[3] GEMS can permanently replace the firmware stored on every AV-TSX in the field. In the case of the AV-OS, GEMS cannot replace the stored AV-OS firmware, but GEMS can cause the running software to be altered. That alteration will persist until the AV-OS unit is powered down and the memory card removed, but it is not permanent.

Section 6.10. However, the Diebold voting system was not designed with those kinds of firewalls in place, and it was not constructed in a way that would provide inherent resistance against the spread of virally propagating malicious code.

4.1.5 Insufficient Controls on Code Integrity

The Diebold devices do not contain strong controls to protect the integrity of their software. For instance, the AV-TSX can be upgraded in any of several ways simply by placing unauthenticated files on a memory card. The GEMS application can be upgraded simply by installing new software on the GEMS PC. All of the software on the AV-TSX and GEMS PC is installed on writable storage (either non-volatile flash memory or a magnetic hard disk). For instance, the bootloader, WinCE operating system, and BallotStation are all stored on writable storage on the AV-TSX and can all be upgraded using authorized channels. The consequence is that if an attacker can ever run malicious code on any of these machines, even once, the attacker can permanently replace all of the software stored on the machines. Moreover, because the upgrade process is under software control, there is no easy way to reset a machine and restore it to a safe state once its software has been corrupted. An AV-TSX or GEMS server, once infected, is very difficult to disinfect with confidence.

This is an architecture that is also shared by commodity PCs. However, it is not an inevitable or necessary property of a computing system. The AV-TSX could have been constructed so that its software was stored on write-once storage (e. g., using PROM or EPROM technology). Such a design would greatly improve the integrity of the machine's code, because a virus or malicious code would have no way to permanently overwrite the device firmware (or at least, the part that is stored on write-once storage). Under such an architecture, the effects of a virus could not persist across reboots, and in particular, the act of powering down a machine between elections would eliminate the virus. An even more aggressive architecture might involve rebooting the machine after every voter, so that any malicious code that made its way into memory while one voter was voting would not be able to persist in memory to affect the next voter. The AV-TSX unit and GEMS application were not constructed in this way, and as a result a virus or security breach in one election can affect every subsequent voter who uses that machine in every subsequent election.

In contrast, the AV-OS provides significantly better controls to protect the integrity of its firmware. The Red Team has informed us that the firmware on the AV-OS is stored on an internal EPROM chip. When the AV-OS is powered up, it loads its software from the internal EPROM chip. While an individual with physical access to the AV-OS unit could overwrite or modify the firmware stored on that chip, malicious software cannot. Consequently, if the running software on AV-OS becomes corrupted with malicious software at any point, rebooting the AV-OS will reload the software from its internal EPROM chip and thus (in the absence of physical tampering) will clear out any malicious software. In other words, even if the AV-OS becomes infected by a virus, the virus cannot install itself permanently on the AV-OS. This is a significant security advantage for the AV-OS.

4.1.6 No Way to Verify Code Integrity

The Diebold AV-TSX and GEMS machines do not provide any secure way for an election official to verify whether the software resident on the machine has been modified. For instance, a cautious election official might wish to occasionally spot-check a random sample of machines to confirm that they have the correct software installed. Unfortunately, GEMS and the AV-TSX provide no way to do that securely.

The AV-TSX does report its version number when it is powered up. However, this version number cannot be trusted. If the device software had been corrupted and overwritten with a malicious replacement, the replacement could simply lie about its version number and report exactly what the legitimate software would report.

Version numbers can be helpful for detecting accidental failures to install the correct version of the software. For instance, if the certified version of the software is version 4.6.4 but the machine reports version 4.5 or 4.7, then one can be certain that the machine is running the wrong version

of software (or, alternatively, that the certified version is so buggy that it fails to report its own version number accurately). However, if the machine reports version 4.6.4, all we know is that either the machine is running the proper software or else it is running improper software that was programmed to misreport its version number. To use an analogy credited to Dan Wallach, if the airport police walk up to a stranger at an airport and ask him "Are you a terrorist?" and he replies "No", have we really learned anything? Similarly, if we ask a voting machine "Are you running malicious software?" and the voting machine's software reports that it is not, there is no reason to trust it. Consequently, while self-reported version numbers may be helpful at detecting accidental misconfiguration or inadvertent error, they are not useful at detecting security breaches [22].

Voting machines can be built that do not have this flaw. For instance, the voting machine could be designed so that its bootloader is stored on write-once storage (PROM or EPROM). The bootloader could make a record of the cryptographic hash of the software that it loads and print that record on the zero tape[4] before the election. Or, the bootloader could contain a public key and could check that the software is properly signed before loading it. Or, the device could use a Trusted Platform Module or other secure hardware technology, such as that standardized by the Trusted Computing Group (TCG). Such a design would allow the operator to verify that the machine's software has not been modified or altered. Neither GEMS nor the AV-TSX has this capability.

The inability to detect malicious software, or verify its absence, makes devising effective defenses against virally propagating malicious code and other kinds of attacks more difficult. Formulating mitigation strategies to detect viruses would be easier if the AV-TSX provided a way to securely verify that the software resident on the machine has not been altered. The absence of this detection capability makes it harder to be fully confident that the voting system has not been subverted and heightens the impact of vulnerabilities that can be used to propagate malicious code virally.

This criticism does not apply to the AV-OS. The AV-OS does not need a way to detect whether its firmware has been corrupted by virally propagating malicious software, because its design already prevents this from happening in the first place, as explained in Section 4.1.5.

4.1.7 Reliance on COTS Software

In several places, the Diebold system relies on commercial off-the-shelf (COTS) software written by third parties. The use of COTS software has significant cost and efficiency advantages for the voting system vendor, as it allows the vendor to reuse existing code and avoid reinventing the wheel. However, this reliance on third-party COTS software also has security disadvantages.

First, GEMS runs on Microsoft Windows and relies on the security of the Windows operating system. Unfortunately, the version of Windows used in the Diebold system has a number of security vulnerabilities. In addition, securing Windows requires keeping the system fully up-to-date on all security patches. Unfortunately, the special circumstances associated with voting systems make it difficult to keep the Windows operating system patched and up-to-date. The Diebold system is tested and certified with a specific version of Windows; changing or upgrading that version might invalidate the certification and may not be permissible. Also, the most common way of keeping Windows machines up-to-date is to connect them to the Internet and have them regularly poll the Microsoft website to see whether there are any security patches to be installed. That approach cannot be applied here: it is not safe to connect the GEMS PC to the Internet at any time. Connecting the GEMS machine to the Internet, even just for a moment to download patches, creates a security exposure.

For all of these reasons, it is likely that the GEMS machine will be running a version of Windows that does not have the latest security patches applied and that is vulnerable to known, published attack methods. This prediction was confirmed by the Red Team's experiments [3]. When the Red Team downloaded a standard attack tool widely available on the Internet and pointed it at the GEMS machine, it immediately identified and exploited a known vulnerability in the Windows installation on the machine. As a consequence, if any malicious or compromised device were ever

[4] When the AV-OS and AV-TSX are powered up on the morning of election day, they print a *zero tape* on their thermal printer. The zero tape is intended to provide evidence that no votes are currently stored on the machine.

connected to the same network as the GEMS machine, then we expect that the device would be able to subvert the GEMS PC. We consider it unlikely that even a careful county system administrator would be able to configure the GEMS PC in a way to prevent such attacks from being successful. This is a risk with relying on commodity software that relies on frequent patching for its security.

Also, the security of GEMS against insider attacks and manipulation by a rogue GEMS operator relies on the security of Windows. Microsoft Windows was not designed with this threat model in mind. First, Windows treats the user (the human operator) as trusted. For instance, Windows does not attempt to prevent the user from executing any program of the user's choice or modifying any file to which the user has access, while GEMS security relies on preventing precisely this type of execution and modification. In particular, GEMS includes access controls as part of the GEMS application, but it does not control what the user can do by interacting directly with Windows or with other Windows applications. This design effectively allows a human operator intent on wrongdoing to bypass GEMS and tamper with the vote tallies in the election database, the election definition files, the GEMS application itself, and other data on the GEMS PC. Put another way, the security properties of Microsoft Windows do not seem in line with the security requirements of GEMS.

Other COTS software used is better aligned with the security goals of the Diebold system. For instance, both the AV-TSX and GEMS use the OpenSSL library for communication security. The OpenSSL library is a de facto industry standard and was an excellent choice for that purpose, as the threat model that OpenSSL was designed to withstand matches precisely what the Diebold voting system needs. However, even OpenSSL has bugs. Hursti [19] notes that security vulnerabilities have been reported in the version of OpenSSL used by the AV-TSX, 0.9.7e. We do not know whether these bugs can be exploited in the context of the Diebold system, but they illustrate the need to keep COTS software up to date.

A second disadvantage of reliance on COTS software is that the use of COTS software makes analyzing and gaining full confidence in the security of the voting software more difficult. Both GEMS and the AV-TSX rely heavily on third-party COTS software: for instance, the AV-TSX uses Windows CE. Diebold did not provide us with the source code to any of this third-party COTS software, presumably because its licenses forbade it. This prevented us from analyzing the COTS software to determine whether it contains any material security risks or whether there may be any deleterious interactions between the COTS software and the vendor-written software. Because the security of the Diebold voting system relies on the security properties of this COTS software, and because we were not provided with the access required to analyze the COTS software, we may not have identified all vulnerabilities related to its use.

4.1.8 Modems and Other Networks

Background: Regional processing The Diebold system supports a configuration, known as regional processing, where the county central location is augmented with several regional return centers distributed around the county. After the close of polls on election night, poll workers transport memory cards, paper ballots, and other election supplies to the nearest regional return center rather than to county headquarters. In counties with many precincts, this reduces the traffic and congestion at any one return center and enables the county to scale its processing of unofficial results on election night. In counties with polling places dispersed over a broad geographical region, regional processing makes it easier for poll workers to return materials without having to drive large distances.

Regional processing allows for hierarchical processing of election returns. At county headquarters, there is a master GEMS machine that is used to coordinate the entire election. Each regional return center has a "client GEMS" installation that is used to read memory cards, accumulate votes from the memory cards, and upload them to the master GEMS machine at county headquarters. Thus, each regional return center would have a GEMS PC as well as one or more AV-TSX machines (used for reading AV-TSX memory cards) and one or more AV-OS machines (used for reading AV-OS memory cards). The master GEMS machine performs the final tabulation, manages the authoritative election data for the entire county, and produces reports and final election results.

The client GEMS installation at each regional return center needs a way to upload data to the master GEMS. How that is accomplished appears to be dependent upon county practices. One supported method is to use a private network (e. g., a dedicated T-1 line or a county intranet). Another possibility is to use modems to communicate over the public telephone network. It appears to be technically possible to use the public Internet or to use virtual private network (VPN) software to establish a communication channel over the public Internet; we did not attempt to determine whether this is permitted by California law. The specific configuration is apparently left to the county's discretion: the system documentation does not appear to contain any prohibitions or limitations on which configurations are supported or allowed. It is our understanding that California election code may restrict which types of connections are permitted.[5]

Background: Modems The Diebold system can be configured to use modems for several purposes:

- AV-OS units contain an internal modem that can be used on election night to transmit election results to GEMS after the polls close. The GEMS server at county headquarters can be connected to a modem bank to receive election results, and the AV-OS memory card can be configured with the telephone number and other information needed for the AV-OS to dial the phone number of the modems at county headquarters. At that point the poll worker is prompted to confirm the uploading of vote tallies, and then they are transmitted over the phone. There appears to be no authentication or encryption of this phone call.
- Similarly, AV-TSX units contain an internal modem that can be used to upload election results to GEMS on election night. The AV-TSX can be configured with the telephone number, username and password to connect to the GEMS server. In contrast to the AV-OS, the AV-TSX software optionally uses SSL to encrypt and authenticate the call. SSL is a standard protocol for secure communications, and it is widely believed to be secure if used properly. The GEMS server is loaded with a public/private key pair as well as a certificate signed by Diebold. The GEMS server provides the certificate to the AV-TSX, and the AV-TSX validates that it is talking to a party with a public key that has been signed by Diebold before allowing the communication to proceed.
- In counties that use regional processing, the client GEMS PC at the regional return center can dial up the master GEMS at county headquarters over the phone and upload election results to the master GEMS for tabulation. This allows county officials to compile unofficial election-night results in a timely fashion. This connection also appears to use SSL to encrypt and authenticate communications between the client GEMS software and the master GEMS.

In all three cases, the GEMS PC must be connected to a modem bank that is connected to the public telephone network. One way to accept modem connections from AV-TSX machines is to enable Microsoft Remote Access Server (RAS) on the GEMS PC. RAS provides a way for clients to dial in and send TCP/IP packets encapsulated using the PPP protocol. It appears that, if modems are used, the GEMS modems will accept phone calls from anyone who knows the right telephone number to dial and can supply the correct username and password.

Our understanding is that once the AV-OS successfully connects to the GEMS modem and once the communication link has been established, the communication protocol between GEMS and the AV-OS is the same as that used when the AV-OS is connected to GEMS by a serial cable. If a memory card is inserted in the AV-OS, the poll worker can upload the election results on that memory card. If a blank memory card is inserted, the poll worker can download election configuration and data onto the memory card over the phone, just as would be done when writing a memory card via a central-office AV-OS unit. As far as we know, these transactions can be initiated only upon poll worker request — GEMS cannot request communication with the AV-OS.

[5] California Elections Code §19250 (f) states: "A direct recording electronic voting system shall not be connected to the Internet at any time." California Elections Code §19250 (h) states: "A direct recording electronic voting system shall not be permitted to receive or transmit wireless communications or wireless data transfers." We did not attempt to determine whether these two provisions would apply to GEMS or to regional processing centers.

Once the AV-TSX successfully connects to the GEMS modem, a TCP/IP session is established and the communication proceeds following the same protocol that is used when a central-office AV-TSX is directly connected to GEMS via Ethernet or serial cable. This protocol also allows a poll worker to download an election database onto the memory card over the phone. As far as we know, these transactions can be initiated only upon poll worker request — GEMS cannot request communication with the AV-TSX.

When regional processing is used, our understanding is that the client GEMS dials the master GEMS, uses PPP to establish a TCP/IP session, and then proceeds to communicate over TCP/IP using a proprietary protocol designed for this purpose.

This understanding of how modem communication works is based upon our examination of the source code and an interview with a Diebold software developer [1]. However, we have not had the opportunity to confirm our understanding by observing the equipment in operation, and it is difficult to be confident in this description from our examination of the source code. Therefore, it is possible that our understanding of this subject is incomplete or mistaken.

Our understanding is that modems are intended to be used for unofficial results. Modems can be used to transfer electronic results, but those electronic results must be checked in some way as part of post-election procedures. That checking could potentially involve comparing the number of votes cast against the number of voters who signed in on the roster sheet; comparing the paper summary tapes printed by the voting machines at the close of polls to the unofficial results in some or all precincts; and/or discarding the results uploaded by modem after election night and subsequently reading in the results from memory cards from scratch.

As far we can tell, it appears that it may be left to county discretion whether and how modems are used. The system documentation does not appear to contain any prohibition on use of modems. We note that there may be legal restrictions associated with use of modems.[6]

Analysis The use of modems and other shared communication networks in the Diebold voting system poses a special risk to security. The system documentation emphasizes that modems should only be used to transmit unofficial results, and implies that this addresses the security risks associated with these forms of networking. However, the situation is more complex than that.

There are two broad categories of risk associated with any kind of networking of voting equipment:

- *Communication security:* Transmitting information over a shared network opens the potential for an attacker to eavesdrop on, tamper with, or disrupt data while in transit. The Diebold system provides appropriate controls to either prevent or detect and recover from these types of threats. Transmitting only unofficial vote results, and only doing so after the polls have closed, ensures that eavesdropping is harmless, as that data will soon be made public in any event. This also ensures that tampering with that data can be detected during the official canvass. Detection does require that election officials diligently compare every vote total printed on every summary tape against the unofficial results or compare every vote total stored on every memory card against the unofficial results, which places an extra requirement on the official canvass, but this burden is manageable. By ensuring that the system has multiple fallback methods for transferring vote totals in case the network is unavailable or has been disrupted by an attacker, the system provides a way to recover from denial-of-service attacks and network failures. Consequently, we agree that the design of the voting system does address this class of risks in a responsible and effective way.

  However, this is not the only risk associated with networking voting systems:

- *Endpoint security:* Connecting a device to a communication network may introduce the potential for an attacker to attack that device and try to subvert or compromise its integrity.

[6] California Elections Code §19250 (g) states: "A direct recording electronic voting system shall not be permitted to receive or transmit official election results through an exterior communication network, including the public telephone system." It is our understanding that the election results transmitted from the regional results center to county headquarters would normally be considered unofficial election results, not official election results.

Whether such attacks are possible depends upon whether the software that interacts with the network contains any exploitable vulnerabilities. However, history has taught us that connecting software of any significant complexity to a network poses a non-negligible risk. The restriction to transmitting only unofficial election results does not help against this class of threats. For instance, when GEMS is set up to receive unofficial election results by modem on election night, this poses a substantial risk that an attacker might be able to dial in, pretend to be a unit in the field, and exploit some vulnerability or configuration error in GEMS or in the modem server or RAS server.

Similarly, in counties that use regional processing, the client GEMS at the regional return center connects to the host GEMS at county headquarters using some unspecified communication link, which could be a modem, a dedicated point-to-point link (e. g., a private T1 line), a connection to the county intranet, a VPN over the public Internet, or even a public Internet connection. Depending on how this is implemented, this may create a possibility for an attacker to subvert this communication link (perhaps as a man-in-the-middle, or perhaps by spoofing one party to the other) and try to compromise one of the two endpoints.

One of the greatest reasons to be concerned about these kinds of attacks is the size of the population of potential attackers who might have the opportunity to mount modem- or network-based attacks. When a voting system is connected to the public telephone network or to any shared network, there is a risk that a hacker anywhere in the world, not necessarily on U.S. soil and not necessarily subject to U.S. law, could attack the voting system from afar. Given the current state of computer security, such attacks might be essentially untraceable: it is often effectively infeasible to trace network-based attacks back to their source. There are many parties who might have both the motive and the opportunity to attack voting systems remotely, and there are few effective deterrents against such attacks.

We can see three strategies for defending against attacks on endpoint security. The first and most natural strategy is to completely avoid use of shared or public networks. This risk avoidance strategy is effective but it denies officials the ability to benefit from the administrative and efficiency advantages of telecommunication networks, so it may or may not be satisfactory. The second strategy is to design software that is impervious to attack. Unfortunately, this is difficult to achieve, and it is difficult to know when you have succeeded. As our analysis of the code illustrates (see Chapter 5), other parts of the code are not vulnerability-free, so there is no particular reason to expect that the code exposed to remote attack is necessarily vulnerability-free. The third strategy is to restrict the adversary's access to the communication link, to make it harder for an attacker to inject malicious data onto the communication channel. For instance, one might use a dedicated point-to-point link or a VPN or use cryptography to authenticate all traffic sent across the communication channel. This is a wise risk reduction strategy. In the case of the Diebold voting system, there are limits to how much one can restrict the attacker's access to these communication channels: because AV-OS and AV-TSX units are sent out to the field and are left unattended without strong physical security, an attacker who is capable of tampering with them while they are left unattended could obtain all of the cryptographic secrets needed to communicate with GEMS. The safest approach is probably to apply all three strategies to the best of one's abilities, including limiting communication over shared networks to the minimum necessary.

A thorough analysis of the risks requires access to working equipment configured in a way that is representative of how counties use the system in practice. We neither had access to working equipment nor information about how communication equipment is normally configured in practice, and the Diebold system documentation provided little guidance on how to configure modems or externally accessible networks to be secure.

Ultimately, we were not able to gain confidence in the security of the Diebold software if it is connected to modems or shared communication networks. The voting system seemed to rely primarily on the unofficial nature of results transmitted over modems for security, but as discussed above, that alone is not sufficient to prevent or detect subversion of the voting system.

4.2 Implementation

4.2.1 Input Validation

Input validation is one of the most important practices that developers of security-critical software must follow. Some experts estimate that approximately half of all software vulnerabilities can be attributed to failure to properly validate inputs from untrusted sources. The best practice is to establish a discipline to ensure that all inputs are validated, for instance, by checking all inputs against a template or whitelist as soon as they are read from any untrusted source and before they are used for any purpose.

We did not find a consistent pattern or discipline of input validation in the source code. Untrusted inputs are occasionally compared against a whitelist or template describing expected values but are more frequently not checked at all. Integers read from untrusted sources are sometimes bounds-checked immediately after being read but sometimes not. Strings are not usually checked for null-termination and are rarely matched against a whitelist or regular expression.

4.2.2 Defensive Programming

Defensive programming is another recommended practice. It involves checking all data provided by other software components just before using the data. Even if one expects that the source of the data has already verified the correctness of the data, each recipient also redundantly checks the data. For instance, pointers are verified to be non-null before being dereferenced, indices are confirmed to be within bounds before being used, and so on. The philosophy is that the program should be constructed to be robust against unexpected inputs and should fail gracefully even if other components contain unexpected bugs.

The use of defensive programming in the Diebold source code was variable. In a few places, the source code was written defensively, carefully checked all inputs, and appeared to be reasonably robust. In other places, the code made unchecked assumptions about the data it used, was not written defensively, and did not appear to be as robust as it could have been. We noticed that the latter appeared more frequently in places where the programmer might not have been expecting malicious or erroneous inputs (e. g., some of the code that handles data read from the election database or other files on the memory card) and in non-core code (e. g., debugging or logging code, code that is used only to print reports, or code for system administration tasks). In some cases, the absence of defensive checks did not lead to any bugs: even though the callee failed to defensively check all necessary preconditions on the inputs, all of the callers happened to establish those preconditions anyway. In other cases, the absence of defensive checks led immediately to security vulnerabilities. In all cases, the absence of local defense against buffer overflows and other error conditions creates a software maintenance hazard: as the code evolves there is a risk that a developer might, without realizing it, add a new call site that violates the implicit preconditions and thereby creates a serious vulnerability.

In many places, the failure to program defensively appeared to be of no particular import. However, in some cases, the failure to program defensively led to serious, exploitable security vulnerabilities. The reason that security engineers often recommend applying defensive programming to all code, not just code that is known to be exposed to an attacker, is that programmers often make unjustified assumptions and fail to anticipate the ways that attackers might be able to provide unexpected inputs. The failure to consistently apply defensive programming techniques probably contributed to the number of exploitable implementation-level vulnerabilities that we found.

4.2.3 Choice of Programming Languages and Libraries

The choice of programming language can have an influence on the frequency of implementation-level vulnerabilities. The Diebold system uses assembly languages, C, and C++. These programming languages are known to be prone to several common types of security vulnerabilities,

including buffer overflows, format string vulnerabilities, and integer overflows. We found instances of all these vulnerabilities in the source code we analyzed.

Many security engineers recommend use of memory-safe, type-safe programming languages, because those languages have inherent resistance to several of the most common types of security vulnerabilities. For instance, until recently, buffer overflows were consistently the number one publicly reported vulnerability [6]. Memory-safe languages, like Java or C#, effectively eliminate buffer overflow vulnerabilities, while programs written in older languages like assembly, C and C++ are known to be at risk for these vulnerabilities. Because it is so easy to make a catastrophic mistake in these older languages without realizing it, many security practitioners recommend that, all else being equal, projects where security is critical should consider using a more modern, memory-safe programming language. We do not mean to suggest that systems written in languages like C or C++ are necessarily insecure. However, programming securely in those languages requires more attention to detail and more experience with secure programming. It appears that the necessary level of care was not taken in the construction of the Diebold voting system.

The use of older programming languages can be partly mitigated by appropriate selection of libraries and other programming platforms. In this respect the AV-TSX and GEMS code can be credited with frequently using safer libraries, which partially reduces the risk associated with their use of the C++ programming language.

For instance, the AV-TSX and GEMS source code often uses the Microsoft Foundation Class (MFC) `CString` class to manipulate strings. This C++ idiom is inherently safer than using C-style strings, because `CString` was designed to avoid many of the common pitfalls associated with C-style strings (e. g., `CString` performs its own memory management and thus tends to prevent buffer overflows). However, the AV-TSX and GEMS code is not consistent in its use of these safer libraries. For instance, it also frequently uses C-style strings, and occasionally uses them incorrectly in a way that creates security vulnerabilities.

The MFC class `CString` provides the function `Format` which works similarly to `sprintf`. Unlike `sprintf`, however, `CString::Format` always ensures it has allocated the memory it needs before writing to its internal buffer. This makes `CString::Format` safe from buffer overflow vulnerabilities and much safer to use than character buffers (however, `CString::Format` does not prevent format string vulnerabilities).

Comments in Diebold's code indicate that developers were aware of the benefits of `CString` over character buffers:

uploaddlg.cpp:340

```
340  Assumes buf is large enough for a token
341  This would be better if it delt[sic] with CStrings
342  rather than with fixed buffers. Gems implemented
343  this improvement at one point.
```

Had the safer `CString` functions been consistently used, nearly all buffer overflow vulnerabilities would have been prevented. For example, in one particular function, both a `CString` and a character buffer are used. The character buffer usage leads to a vulnerability while the `CString` usage does not. See private appendix Issue 4.1 for more information.

In contrast, the AV-OS uses the standard C libraries and so remains susceptible to all of the risks associated with the C programming language.

4.3 Engineering Practices

Our interview with Talbot Iredale [1] provided useful insight into Diebold's general software engineering practices. Overall, Diebold's practices seem to be similar to those of most small- to medium-sized software development firms. These practices may be sufficient for ordinary commercial software, but they are inadequate for meeting the rigorous security requirements of voting software.

No Formal Threat Model or Security Plan The first step in designing a secure system is to write a formal *threat model*, a document which clearly identifies the assets that the system must protect and the threats that the system could face. A threat model also attempts to catalog the potential types of attackers, their capabilities, and their likely motivations. The goal of a threat model is clarify the system's security requirements so that there will be a basis for evaluating the security of potential designs of the system. We present a generic threat model for large electronic voting systems such as the Diebold system in Appendix A.

Once a threat model has been devised, the next step is to design a security plan aimed at countering the potential threats to the system and to evaluate its strengths and weaknesses in light of the threat model. The security plan should be a formal document and should clearly state how it deals with each of the threats that have been identified.

In our interview, Mr. Iredale stated that Diebold has neither a formal written threat model nor a formal security plan for its voting systems. Indeed, we found no evidence in the source code that systematic analysis of threats had been performed. Instead, the security measures that are in place appeared to be *ad hoc*. For example, the same threat often receives inconsistent treatment: the AV-TSX uses cryptography to protect some of the data on its memory cards, but the AV-OS does not — even though the threats to both types of memory cards are largely the same. Similarly, GEMS restricts what an operator can do through the GEMS user interface, but the system does not effectively control what an operator can do using the underlying Windows interface.

No Formal Security Training Diebold has about 25 developers that work on electronic voting systems, including those who focus on documentation, testing, and hardware. When new developers arrive at the company, they do not receive any kind of formal security training. Mr. Iredale states that some developers have security backgrounds but no one is dedicated to handling security issues. They have two small groups of quality assurance testers of approximately four people each, but none of them are dedicated specifically to security or red-team testing.

Weak Source Code Review Process Diebold uses standard versioning software (CVS) to manage the development of their source code. Any developer can check code into CVS and the code is not reviewed by other developers before it is committed into the repository. Mr. Iredale states that every CVS check-in causes an e-mail to be sent to developers who are responsible for reviewing the code. Initially, they do "random checks" on most of the code and do a "closer review" of the more critical portions. Although Mr. Iredale claims that 100% of the code is reviewed by another Diebold employee within a few weeks, there seems to be no formal procedure for assigning code to other employees for review. It seems possible that, without formal procedures, some source code could remain unreviewed before release. Issue 5.2.24 suggests that this is the case.

No Unit Testing or Red Team Testing There is no formal requirement to develop a set of unit tests that correspond to each piece of code checked into CVS — the option of doing this is strictly up to individual developers. The testing group will later check for correctness based on standard test plans.

This manual testing methodology seems fragile and inappropriate for security-critical software. Often, a developer who modifies one part of the code will inadvertently break another part of the code. It would be very difficult for individual manual reviewers and testers to reliably notice these latent errors. Unit tests written concurrently with the actual source code by the initial developer would be a more thorough strategy. The developer can better construct a set of unit tests that check boundary conditions, pre-conditions and post-conditions of each block of code she writes. These unit tests can then be run systematically at any time by anyone to check for errors. Unit tests should be combined with other automatic systems-level tests that are run regularly when code is checked in.

Diebold also lacks any formal procedures for "red team" testing, where the testers play the role of attackers and attempt to break into the system. This type of testing can detect different types of bugs that "white box" unit and system tests might not catch, such as illegal input handling and failure recovery.

Our analysis has led us to conclude that the design and implementation of the Diebold software does not meet the requirements for a security-critical system. We identified a number of systemic issues that were pervasive throughout the source code or that reflected flaws in the design of the voting system. We also found that the Diebold system's code fails to consistently follow sound, generally accepted engineering practices for secure software. Moreover, we have determined that these systemic weaknesses led to specific flaws that can be exploited by attackers (see Chapter 5).

Fixing these specific flaws without addressing the underlying systemic weaknesses that caused them is unlikely to render the systems secure. Systems that are architecturally unsound tend to exhibit "weakness-in-depth"—even as known flaws in them are fixed, new ones tend to be discovered. As a result, we must conclude that without sweeping changes to its architecture and to the practices under which it is developed, the Diebold system will likely continue to pose a risk to election integrity.

CHAPTER 5

Selected Specific Issues

In this chapter, we detail specific weaknesses we found in the AccuVote-OS, the AccuVote-TSX, and the GEMS election management systems. We discuss the issues, the requirements to exploit the issues, their implications on system security, and the observations that led us to our findings. This chapter is by no means a complete catalog of issues that might exist on these systems. Exact references to specific problems, such as source code excerpts and line numbers, are included in the private appendix.

5.1 AccuVote-OS

There are two types of Diebold AccuVote-OS machines: the AV-OS Precinct Count and the AV-OS Central Count. Since the AV-OS Central Count machine is not used in the field, the attack surface for the Central Count machine is significantly smaller than that of the AV-OS Precinct Count machine. The risk of attack for the AV-OS Central Count is diminished because it is physically located at the county headquarters and not used to process memory cards from the field. Therefore, we focused our efforts on the AV-OS Precinct Count machine. The following comments about the AV-OS machine pertain only to the AV-OS Precinct Count machine.

Three published reports previously exposed serious security vulnerabilities in the AV-OS:

1. Hursti first described critical AV-OS vulnerabilities in his July 2005 report, "Critical Security Issues with Diebold Optical Scan Design" [17]. He analyzed an earlier version of the AV-OS (1.94w) than we studied in this report (1.96.6).

2. A study entitled "Security Analysis of the Diebold AccuBasic Interpreter" [33] was published in February 2006 by Wagner, et al. The authors were charged by the California Secretary of State with reviewing the security implications of the AccuBasic subsystem of the AV-OS and AV-TSX in response to Hursti's findings. The study covers the same version of the source code that we were provided for this study.

3. Kiayias, et al. discovered new vulnerabilities in the AV-OS in October 2006 and reported their findings in "Security Assessment of the Diebold Optical Scan Voting Terminal" [23]. The study was supported by the Office of the Connecticut Secretary of the State. They were able to find these security holes solely by experimenting with the machine and without any access to the AV-OS source code.

The combination of security vulnerabilities in these three reports provides an attacker with numerous vectors to breach the integrity of an election run on AV-OS machines. We confirmed that many of these previously reported holes still exist in the current version of the AV-OS source code. Because of the highly exploitable nature of these bugs already found and the short time frame of this study, we spent less time looking for new attacks on the AV-OS itself and instead concentrated our efforts on other areas of the system. Our other findings, especially those concerning the

GEMS server, present additional vectors for attacking the AV-OS that build on these existing vulnerabilities.

In this section, we describe the problems we confirmed about the AV-OS Precinct Count machine.

Issue 5.1.1: *Data on the AV-OS memory cards is unauthenticated.*

The AV-OS does not use any strong authentication mechanisms to confirm that the data on the inserted memory card originates from a legitimate source. The machine always assumes that the memory card data is trustworthy and does not sufficiently validate the data before use. Thus, an attacker who can arbitrarily write to the memory card can easily modify its entire contents without being detected.

Each memory card contains the full state of the current election for that machine. All data on the memory card is stored in unencrypted form (other than the supervisor PIN[1], which while obfuscated, can easily be deciphered — see Issue 5.1.8). The memory card contents include:

- Memory card header: firmware revision number, card size, election status, counting mode (absentee or not), master card copy password, global counters (number of total ballots counted, number of election uploads, number of ballot tests, etc.)
- Election header: voting center name and number, download version number, obfuscated supervisor PIN, election title and date, election type, party code table, election configuration flags, and data checksums
- Election definition and data: lists of precincts, ballot cards, races, candidates and voting positions; race counters and candidate counters
- Audit log
- Compiled AccuBasic scripts
- Memory card heap

In order to take advantage of this vulnerability, an attacker would need to have write access to the memory card. One way to gain write access is to have temporary physical control of the card. According to the Diebold AccuVote-OS Hardware Guide documentation [9], the AV-OS memory card is stored behind a locked retaining door. The lock can easily be picked in a short time using paperclips [23]. The memory card is sealed in the machine by procedure, but attacks may be able to bypass the seal to access the card without detection. Once past the seal, the memory card can be removed from the machine and modified. To modify the card, the attacker needs an Epson RBC compatible memory card reader-writer device, which is publicly available for purchase [17]. Another possibility is for the attacker to bring a new Epson smart card from home that contains his arbitrarily-chosen election data. The attacker-controlled card can be inserted into the machine.

An election insider or an intruder who gains temporary access to the machine before or after the election could exploit this vulnerability. A well-prepared attacker would probably need only a minute or two to swap the AV-OS memory card. It is unlikely that this attack could be conducted by a voter on election day. This particular avenue of memory card access takes time, would look suspicious, and would be easily visible to others, since the AV-OS has no physical privacy screens and is often in plain view of both poll workers and other voters.

Another way to obtain write access to the memory card is through GEMS or a GEMS-impersonator. This avenue would not require physical access to the memory card. See Issue 5.1.2 for the details of this attack.

Three implemented mechanisms attempt to detect tampering of the memory card data: memory card checksums, the internal audit log, and the memory card signature. None of these mechanisms provide adequate protection and are easily be subverted by an attacker.

[1] The supervisor PIN is used by election officials to access machine setup functionality.

The primary problem is that all of these mechanisms store their values on the memory card, but an attacker can modify these values to conceal the attack. Each mechanism is described in further detail below in Issue 5.1.3, Issue 5.1.4, and Issue 5.1.5.

The lack of authentication on the AV-OS memory card data is a fundamental security flaw that cannot be overstated. This vulnerability is a stepping stone to highly exploitable attacks that make it much easier for an attacker to compromise the integrity of the election. See Issue 5.1.6, Issue 5.1.7, Issue 5.1.9, Issue 5.1.10, and Issue 5.1.11.

Issue 5.1.2: *The connection between the GEMS server and the AV-OS is unauthenticated.*

When the AV-OS is booted with an empty memory card inserted, the machine will prompt the user to initialize the card by downloading data from the GEMS server. The connection channel between GEMS and the AV-OS, which can be established over either a serial link or a modem line, is neither encrypted nor authenticated. An attacker can learn or reverse engineer the protocol that GEMS uses to communicate with the AV-OS. He can subsequently impersonate a GEMS server to the AV-OS to write into important memory card data fields.

If the memory card is not initially empty, the attacker can first use Issue 5.1.8 to learn the supervisor PIN to reach the option of clearing the existing memory card. A subsequent reboot will prompt the attacker to connect to GEMS.

Neither the AV-OS nor the GEMS server requires any form of authentication token (e. g., a password) in order to connect. An attacker can connect a GEMS clone (e. g., a laptop implementing the reverse-engineered protocol) directly using either a serial cable or a modem. The attacker can also arbitrarily change the phone number that the AV-OS uses to connect to GEMS again using Issue 5.1.8. Once connected, the attacker can control:

- The entire election header
- The entire election definition and initial data
- The compiled AccuBasic script

Writing to the memory card by impersonating GEMS is slightly weaker for the attacker than exploiting Issue 5.1.1. Whereas the previous vulnerability allows writing to any bit of the memory card, the attacker here cannot write to the memory card header, the audit log or the card's memory heap section. Still, it is possible for the attacker to use this exploit to compromise the entire machine. See Issue 5.1.9, Issue 5.1.10 and Issue 5.1.11.

Even if the connection between GEMS and the AV-OS were securely authenticated, it would still be possible for an attacker with access to GEMS to compromise the AV-OS. No longer would the attacker be able to impersonate the GEMS server, but vulnerabilities we found in GEMS could allow an attacker to modify the originating memory card data. See Issue 5.3.2, Issue 5.3.4 and [3].

After the election, the AV-OS connects again to the GEMS server to upload the results of the election. Again, no encryption or secure authentication is used. This allows an attacker to impersonate a legitimate AV-OS to the GEMS server. An imitation AV-OS would be able to upload bogus results to the GEMS server if the legitimate AV-OS had not already uploaded results. When the legitimate AV-OS attempts to upload, the GEMS server will reject the results as already uploaded.

Issue 5.1.3: *The memory card checksums do not adequately detect malicious tampering.*

The AV-OS uses non-cryptographic checksums to detect errors in the memory card, but the checksums are weak and do not protect against malicious tampering. The checksums are stored on the memory card with the data. The checksum mechanism does not provide a cryptographic guarantee of either data integrity or authenticity. It is only effective to detect non-adversarial changes to the card contents, such as transmission errors or hardware faults. A malicious attacker who modifies data on the card can easily calculate a new valid checksum to avoid detection.

In the election header of the memory card, there are eleven different 16-bit checksums values that cover most, but not all, of the data fields on the card. The checksums are checked immediately when the machine is turned on, when the machine enters pre-election mode, and when the machine enters post-election mode. During the election, the checksums are updated as ballots are processed, but the checksums are not checked.

The eleven checksums are divided into three categories:

- Header checksums covering: the election header, precinct headers, precinct cards, ballot card headers, contest (race) headers, candidate headers, and ballot card voting positions
- Counter checksums covering: contest (race) counters, candidate counters, and ballot card counters
- A text checksum covering (most) strings on the memory card, such as the name of the voting center, the election date, and the names of races and candidates. This does *not* include the compiled AccuBasic script or the audit log.

Although there is an allocated slot in the election header for a twelfth checksum named `auditLogChecksum`, it is never assigned or accessed. We assume the audit log checksum was planned for but never implemented.

The header and counter checksum algorithms are very primitive. For each checksum, the machine calculates the arithmetic sum of the corresponding data members on the memory card. For example, the candidate counter checksum is the arithmetic sum of the values of each candidate's total number of votes. If Alice has 36 votes and Bob has 23 votes, the candidate counter checksum is 59.

The text checksum algorithm is slightly more complex but still insecure. An integer counter is first initialized to zero. For each character of each string covered, the counter is incremented and the bits of the character are XORed with the integer counter. These XORed values are then arithmetically added together to create the text checksum. We can more precisely express this algorithm using the following pseudocode. (Note: This is *not* an except from the actual source code, but rather a simplified description of what it does written in an imaginary programming language.)

```
def text_checksum():
  int checksum = 0;
  int counter = 0;
  for each string s:
    for each char c in s:
       checksum += c ^ counter;
       counter++;
  return checksum;
```

The checksums are only effective for detecting accidental errors in the memory card data. The checksums are completely ineffective against a malicious attacker attempting to tamper with the card data fields. An attacker can easily change the checksum stored on the card to reflect the changes made to the data. Moreover, an attacker can simply ignore the checksum if he makes two or more changes that offset one another. See Issue 5.1.9 for an example of this.

The checksums are all generated locally by the AV-OS machine. Initial checksum generation happens after the memory card contents are downloaded from GEMS — the checksum is not sent with the data by GEMS. If there are transmission errors between GEMS and the AV-OS, the checksum will be calculated on the erroneous memory card data. Similarly, if a man-in-the-middle attacker tampers with the link between GEMS and the AV-OS, the checksum provides no defense.

Issue 5.1.4: *The audit log does not adequately detect malicious tampering.*

The audit log is another security measure that the AV-OS uses to try to detect machine abnormalities. Under normal operation, all major events that occur on the machine are recorded in the audit log. Each event transaction records the event type and the event time. Only two types of events, audit log initialization and memory card insertion, record the date as well. In all, there are 30 different types of log transaction events. The log has capacity for 512 transaction entries and wraps around to overwrite the first entry when the log reaches capacity. It is stored in plaintext on the memory card.

Like the checksums, the audit log is stored on the memory card and can be modified if the attacker has control of the card (see Issue 5.1.1). The attacker can arbitrarily add, modify, or remove log transactions so that the log will be consistent with the other data on the memory card after an attack. Thus, the audit log is completely insecure and should not be trusted by election authorities as evidence that no attack has occurred.

Issue 5.1.5: *The memory card "signature" does not adequately detect malicious tampering.*

The AV-OS uses a struct called `MemCardSignature` to check whether the memory card was switched while a ballot card was being scanned. The `MemCardSignature` is not a signature in the cryptographic sense, but rather a weak check on the consistency of three counters on the memory card. It consists of three integer counter values: the total number of ballots, the number of absentee ballots, and the number of non-absentee ballots counted so far. The sum of the latter two values should equal the first value. The signature covers no other parts of the memory card. The signature is read from the memory card immediately before a ballot is scanned and the saved signature is checked against the signature of the memory card immediately after the scanning is complete. If the signatures differ, counting is aborted and the machine halts.

Since all three counters are stored on the memory card, this vulnerability allows an attacker to swap the memory card in the AV-OS with his own malicious memory card as a ballot is being scanned. For example, if the attacker is the first voter of the day, the three counters would all be zero. As long as the new memory card has the same signature (i. e., counter) values, the attacker can modify other parts of the memory card (e. g. the number of votes for each candidate) without detection.

Issue 5.1.6: *Buffer overflows in unchecked string operations allow arbitrary code execution.*

There are at least four buffer overflow vulnerabilities in the code used to upload election results to the GEMS server. When the machine prepares the memory card data to be sent to GEMS, it uses the unsafe `sprintf` function to write data to a buffer string `buf`. When one element of the format string argument to `sprintf` is a string that comes from the memory card, an attacker may be able to overflow `buf` on the stack. If the attacker controls the memory card (see Issue 5.1.1), he can delete the null-termination character of the string element to extend its length to be larger than the size of `buf`. The code here is not defensively-programmed and incorrectly assumes that the strings on the card are never longer than expected.

We believe that each of these vulnerabilities can be used by an attacker who controls the contents of the memory card to execute arbitrary code on the AV-OS. One of the four overflows occurs in the code immediately before uploading the memory card's election header to GEMS. If the attacker overflows `buf` and takes control of the machine at this point, he can modify both the election results on the card and the results to be uploaded to GEMS.

See private appendix Issue 5.1.6 for more information.

Issue 5.1.7: *Integer overflows in the vote counters are unchecked.*

The AV-OS does not check for integer overflow in the counters that keep track of candidate votes. Each candidate vote counter is a 16-bit unsigned integer, which can hold up to 65,535 votes. If the vote counter reaches its maximum value and more votes are added, the counter

will wrap around to zero and continue counting. The counter will overflow without warning. Because of the unchecked overflow, a large counter value is equivalent to a small negative counter value. For example, a counter value of 65,526 is equivalent to a counter value of -10 because adding 10 votes to either value results in a counter value of 0.

This vulnerability could allow an attacker to undetectably switch a predetermined number of votes from one candidate to another. This attack was first demonstrated by Hursti [17], and we confirm that the AV-OS remains vulnerable. To see an example of this attack, see Issue 5.1.11.

Issue 5.1.8: *The machine does not adequately protect the supervisor PIN.*

The PIN used by election supervisors to administer an election is stored in obfuscated form on the memory card. The obfuscation procedure is a linear function that can easily be reversed without special knowledge. This was first shown by Kiayias et al. [23], who had physical access to an AV-OS but no access to the source code. The obfuscated form obf of a supervisor PIN pin is as follows:

$$\mathsf{obf} = \mathrm{encode}(\mathsf{pin}, \mathsf{key}) = \mathsf{pin} + (\mathsf{key} \times \mathsf{MAGIC}_1) + \mathsf{MAGIC}_2$$

where MAGIC_1 and MAGIC_2 are hard-coded 20-bit magic constants. The value key[2] is stored in cleartext at a fixed location on the memory card adjacent to obf. To recover pin, an attacker could compute:

$$\mathsf{pin} = \mathrm{decode}(\mathsf{obf}, \mathsf{key}) = \mathsf{obf} - (\mathsf{key} \times \mathsf{MAGIC}_1) - \mathsf{MAGIC}_2$$

Anyone with access to the machine for a few minutes can reverse-engineer both the magic constants and the formulas used to encode and decode the PIN [23]. The same magic constants are apparently used on every AV-OS machine.

This privilege escalation vulnerability allows anyone with read access to the memory card to learn the supervisor PIN. An attacker can use the PIN to access supervisor functions, such as changing system setup parameters (the GEMS dial-up phone number, feeding options, etc.), duplicating the memory card, or clearing the memory card.

See private appendix Issue 5.1.8 for more information.

Issue 5.1.9: *Votes can be swapped or neutralized by modifying the defined candidate voting coordinates stored on the memory card.*

The Kiayias report identified an attack against the AV-OS that can be mounted by anyone who can tamper with the AV-OS memory card [23]. The memory card contains a map of the layout of the paper ballot, listing for each candidate the location where that candidate's bubble can be found. A mark in that location will be counted as a vote for that candidate. Locations are identified by $(row, column)$.

The report points out that it is possible to modify those locations on the memory card to ensure that marks for a candidate will not be counted. For instance, if the attacker modifies the voting coordinates for a particular candidate to point to some other location on the ballot where no marks are likely to occur, then it is likely that the machine will never detect any votes for that candidate.

As the Kiayias team discovered, the checksum on the contents of the memory card does not prevent this attack. For example, if the attacker modifies a candidate's voting coordinates from (*row*, *column*) to ($row - 1$, $column + 1$), the ballot card voting position checksum stays constant. Their report also reveals that it is possible to swap two candidates, so that a mark

[2] As an aside about code quality, the key used in the above pseudocode is actually named `publicKey` in the source code. It is a 16-bit unsigned integer stored in the election header. It is clear that `publicKey` does not refer to a cryptographic public key as computer security experts would use the term. Rather, it is just a random nonce used in the obfuscation code that resides on the memory card. The terminology used is non-standard and can be confusing or misleading.

for Smith is counted as a vote for Jones and vice versa. In general, an attacker who can tamper with the contents of the memory card can rewrite the map of the ballot and cause ballots to be miscounted by the AV-OS in a controllable way.

We confirmed that this vulnerability is present in the version of the AV-OS source code that we examined. The Red Team has also confirmed this attack [3].

Issue 5.1.10: *Multiple vulnerabilities in the AccuBasic interpreter allow arbitrary code execution.*

Diebold uses a scripting language called AccuBasic to customize the format of reports printed on the AV-OS, such as the Election Zero report (printed before the election) or the Election Results report (printed after the election). An AccuBasic script is stored on the memory card, and the AV-OS software interprets the script from the card. An earlier study commissioned by the California Secretary of State revealed that the AccuBasic interpreter software on the AV-OS contains security vulnerabilities that allow a malicious AccuBasic script to compromise the integrity of the AV-OS [33]. In particular, a carefully crafted AccuBasic script can, when it is executed, exploit bugs in the AccuBasic interpreter to inject malicious code into the AV-OS and then begin executing that malicious code. Consequently, an attacker who can tamper with the AccuBasic script can take control of the AV-OS machine and cause it to misbehave in any way that the attacker chooses (e. g., misrecording votes).

We confirmed that the flaws discovered earlier exist in the AV-OS (we examined the same software that was examined earlier). We summarize briefly the impact of these vulnerabilities. See the earlier report for full details [33].

The AccuBasic script on the memory card is written by GEMS. If an insider with access to GEMS is malicious, he could replace the legitimate AccuBasic script with a malicious script to take over the machine. This means that the security of GEMS is critical: any compromise of its security could affect every AV-OS machine in the field.

Also, it would be possible for an individual with unsupervised access to the memory card to tamper with its contents and introduce a malicious AccuBasic script. Because the data on the memory card is not cryptographically protected (see Issue 5.1.1), anyone who has access to the memory card can modify its contents or swap it for a prepared "evil twin" card. That could allow someone who had unsupervised access to the memory card to compromise the software on the AV-OS machine that it is inserted into.

This vulnerability could help a virus to spread. If GEMS is infected, it can infect every AV-OS memory card that is written. Inserting an infected memory card into an AV-OS machine will allow execution of malicious code on the AV-OS once the operator prints any report. Also, any individual who has physical access to a memory card can infect it. This vulnerability alone is not enough to create a virus that spreads from AV-OS to AV-OS, but it could be one building block that a virus writer might use, in conjunction with other vulnerabilities. See Section 3.1 for further details.

Issue 5.1.11: *A malicious AccuBasic script can be used to hide attacks against the AV-OS and defeat the integrity of zero and summary tapes printed on the AV-OS.*

We confirmed that many of the vulnerabilities identified by Hursti in his July 2005 report [17] remain present in the current version of the AV-OS.

For instance, Hursti identified an attack that combines integer overflows (Issue 5.1.7), memory card tampering (Issue 5.1.1), and a malicious AccuBasic script to cause the AV-OS to transfer votes from one candidate to another. We briefly outline the attack here. Suppose that the attacker wants to fraudulently transfer ten votes from Brown to Smith. The attack proceeds as follows:

1. The attacker sets all the vote counters on the memory card to zero, except that Brown's counter is set to $65,526$ (namely, -10) and Smith's is set to 10. The ballot box has now been pre-stuffed: Brown starts out at a disadvantage, and Smith starts out at an advantage.

2. The attacker writes a malicious AccuBasic script onto the memory card. This script provides a custom format for the Election Zero report so that the tape will show zero votes for Brown and Smith even though the vote counters on the memory card are non-zero. However, the script for printing the Election Result report is left unmodified, so that the report accurately prints the values of the vote counters on the memory card.
3. When the AV-OS is powered up on election morning with this memory card inserted, it will print a fraudulent Election Zero report showing all zeros (even though the memory card's electronic ballot box has been pre-stuffed).
4. As voters vote, the vote counters will be incremented accordingly. On the 10th vote for Brown, the vote counter for Brown will overflow and wrap around and become zero, though there will be no way for anyone to notice that this has happened.
5. When the polls are closed at the end of the day, the AV-OS will print a Election Result report showing the contents of the vote counters on the memory card. If voters cast 56 true votes for Brown and 44 votes for Smith, the memory card will show 46 votes for Brown (because it was pre-loaded with -10 votes for Brown) and 54 votes for Smith, so the report will incorrectly show that Smith is leading Brown 54 to 46. When poll workers compare the number of voters who signed in against the total number of voters in that contest, they will not find any discrepancies, since this attack did not alter the total number of votes cast: it only shifted votes from Brown to Smith.
6. When the memory card is returned to county headquarters, GEMS will read the (fraudulent) tallies on the memory card, namely, Smith: 54, Brown: 46. When county officials perform the official canvass, there will be no discrepancies between the Election Zero report, Election Result report, and unofficial electronic tallies.
7. If the paper ballots in this precinct are selected to be manually counted during the 1% manual tally or during a 100% manual recount, the fraud will be revealed. However, this is the primary way that the attack can be discovered. If attackers try to tamper with the results in hundreds of precincts, it is likely that the 1% manual tally will detect the fraud in at least one precinct. However, if attackers only tamper with a few polling places, the attack is unlikely to be detected.

This attack requires the ability to make unauthorized changes to the contents of the memory card. Anyone with unsupervised physical access to an AV-OS memory card could attack the machine in this way (see Issue 5.1.1). A malicious GEMS operator could make these changes. Likewise, a security breach/infection of the GEMS PC could allow someone to mount this kind of attack (see Issue 5.1.2).

Hursti also identified the possibility for malicious AccuBasic scripts to print malicious messages to the LCD display or cause the machine to crash under certain circumstances. Those attacks remain possible but appear to have less severe consequences.

Issue 5.1.12: *The physical paper ballot box deflector is under software control.*

The Diebold AV-OS Hardware Guide documentation [9] states that the physical ballot box has two compartments: a primary compartment and a secondary compartment. There is a physical ballot card deflector that determines into which compartment a ballot card is deposited after it is scanned. The direction of the deflector changes depending on programmable sort conditions that are set for each election. For example, ballots with write-in candidates can be deflected into the secondary compartment so election officials can tally these manually after the election closes.

An attacker who controls the machine (see Issue 5.1.1 and Issue 5.1.2) can arbitrarily move the deflector for each ballot card. This can potentially disenfranchise voters if, for example, ballots with write-in candidates are not deflected to the secondary compartment for manual review. Voter privacy can also be affected by using the secondary bin to single out ballots of interest.

5.2 AccuVote-TSX

These are some of the specific issues we identified in the AccuVote-TSX system:

Issue 5.2.1: *The AV-TSX automatically installs bootloader and operating system updates from the memory card without verifying the authenticity of the updates.*

The AV-TSX includes a software update mechanism that allows new bootloader and operating system software to be installed from a memory card inserted into the machine. When the machine boots, it searches the memory card for specially named files. If it finds files named `eboot.nb0` or `nk.bin`, it replaces the bootloader software or Windows CE operating system image, respectively, stored in its internal flash memory with the contents of the files. These filenames have been previously published on the Internet [18].

Hursti was the first to discover that the AV-TSX has no mechanism for checking the authenticity of these software updates [18]. While the machine does employ a simple checksum to make sure the files have not been garbled in transmission, it fails to utilize a digital signature or other mechanism that would prevent an attacker from using the software update feature to install malicious code.

This means that an attacker can create malicious software updates containing arbitrary code, and the software update mechanism will not be able to distinguish these from legitimate upgrades. An attacker who has temporary physical access to a memory card — or control of any machine into which a memory card is inserted — can place his own malicious software update files on the card, and this software will be installed on any AV-TSX machine that is booted with that card in place.

Alternately, an attacker with unsupervised physical access to the machine for as little as a minute could replace the installed memory card with one containing a malicious software update prepared earlier, boot the machine to install the update, and then reinsert the original memory card. This attacker would need to bypass the lock on the memory card door, but we were notified by the Red Team that this can be accomplished quickly using only a ball-point pen. Tamper-evident seals might be used to deter attacks, but creating a seal that requires a sophisticated attack to defeat remains beyond the state of the art [20].

Furthermore, the AV-TSX machine installs bootloader and operating system updates without asking the local user for confirmation. While it does display a message indicating that an update is taking place, this message is displayed in a small font and could easily be overlooked by poll workers. The lack of confirmation increases the odds that this issue could be used to spread voting machine viruses, which we discuss in Section 3.1. The machine does not maintain a log of software updates installed via this mechanism.

Feldman, et al. demonstrate how an identical update mechanism used by the AccuVote-TS model DRE could be used to spread malicious software that would alter the electronic vote records and totals [14]. We believe that similar attacks are possible on the AV-TSX.[3]

One mitigation strategy that has been suggested in the past is to seal the memory card inside the voting machine before the machine is delivered to the polling place. This might make it more difficult for an attacker to access the memory card without being detected. However, it also ensures that the memory card will be installed in the machine when the machine boots. As a result, if the central office AV-TSX that initializes the memory card is compromised so that it stores a malicious software update on each card while initializing it, the AV-TSX machines in the polling places will automatically install the malicious update when they boot on election day.

[3] Diebold has removed another service feature that has been criticized by previous reports [18, 14]. In older versions of the AV-TSX, the system would boot into Windows Explorer instead of the BallotStation application if a file named `explorer.glb` was present on the memory card. This would allow an attacker with physical access to the machine to install malicious software manually. Since this feature has been disabled, attackers would need to exploit issues such as Issue 5.2.1 or Issue 5.2.2 to access Windows Explorer.

Issue 5.2.2: *The AV-TSX automatically installs application updates from the memory card without verifying the authenticity of the updates.*

A second software update mechanism operates after the AV-TSX boots into Windows CE. The Windows kernel launches a program called `taskman.exe`, which eventually starts the BallotStation application. Before running BallotStation, `taskman.exe` searches the memory card for files with the extension `.ins`. These are software update packages in a Diebold proprietary format. Each `.ins` file contains instructions for installing one or more files onto the machine's file system, along with the data to be placed in those files. (The machine's root file system, \, is backed by RAM, so its contents disappear when the machine reboots. The internal flash memory is mounted in a subdirectory called \`FFX`, and removable memory cards are mounted in a subdirectory called \`Storage Card`.)

Like the bootloader-based update mechanisms described above, the `.ins` update mechanism does not attempt to verify the authenticity of the updates, and it does not maintain a log of software updates that have been performed [18]. Unlike the other mechanisms, it does ask the user to confirm each update by touching a button on the screen. However, the system trusts update files to accurately describe their contents. Malicious updates could inaccurately describe their purpose, possibly fooling operators into consenting to their installation.

Attackers could use this mechanism to install arbitrary code onto the voting machine, such as by replacing the BallotStation executable file with an altered version. They could also replace other files on the system or destroy data by replacing it with garbage. Conducting such attacks would require the ability to write the update file onto a memory card as well as the operator's consent; however, Issue 5.2.3 describes a way to avoid the need for consent.

Issue 5.2.3: *Multiple buffer overflows in `.ins` file handling allow arbitrary code execution on startup.*

This issue was first reported by Feldman, *et al.* in their study of the AccuVote-TS [14]. The code in `taskman.exe` responsible for installing the application updates described in Issue 5.2.2 contains multiple buffer overflows in the way it parses `.ins` files that could allow an attacker to execute arbitrary code. For example, a malicious `.ins` file could modify files in the machine's filesystem or launch programs, and it could do these whether or not the user consented to its installation.

To exploit these previously-known vulnerabilities, an attacker would need the ability to write files onto the memory card. Since the machine does not verify the authenticity of `.ins` files, no secret keys would be required.

See private appendix Issue 5.2.3 for more information.

Issue 5.2.4: *Setting a jumper on the motherboard enables a bootloader menu that allows the user to extract or tamper with the contents of the internal flash memory.*

As previously disclosed by Hursti, the motherboard inside the AV-TSX contains a jumper header marked "Debug" [18]. When a jumper is installed here, the machine's bootloader displays a service menu over its serial port when the machine boots. A feature of the service menu called the "mini monitor" allows arbitrary memory locations to be read or written over the serial port. Since the system maps its internal flash memory into the address range 0xA0000000–0xA4000000, the mini monitor can be used to extract or alter the data stored there.

To access this feature, an attacker would need physical access to the inside of the voting machine. The plastic case of the AV-TSX can be removed by unscrewing a small number of screws. After opening the case, an attacker would need to close the jumper circuit by placing a paperclip or small wire against the marked terminals on the motherboard while booting the machine [3]. To interact with the menu, the attacker would need to attach another computer to the machine by connecting a serial cable to the "VIBS Keypad" port (in the rear of the machine). Broken security seals might provide some evidence of the attack, but the machine does not maintain any log files that would show that the service menu was accessed or which commands were issued.

An attacker could use the mini monitor to alter any part of the voting machine's software, including the bootloader, operating system, and BallotStation application. The mini monitor can also be used to create a perfect copy of the internal flash memory, including the software that runs the machine, records retained from past elections, and cryptographic keys stored there. This would be useful in constructing future attacks. Reading or writing the entire flash memory using this method would require several hours of continuous access to the machine, but an attacker could install small software changes or read short memory items like keys in seconds. This method could be used to extract the secret keys from the machine, facilitating other attacks that rely on modifying signed or encrypted files.[4]

Issue 5.2.5: *Keys used to secure smart cards and election data are not adequately protected.*

The AV-TSX uses security keys for various security purposes, including authenticating election definition files, encrypting and authenticating ballot result files, validating voter smart cards, and generating ballot serial numbers. Older Diebold machines used hardcoded keys set when the software was compiled. The version of the AV-TSX that we studied retains those hardcoded keys but also allows county election officials to change the keys that the machine uses. Officials can set three keys, the 64-bit *Smart Card Key*, the 16-bit *Smart Card Magic Number*, and the 128-bit *Data Key*. (Internally, these are labeled `SCKey`, `SCMagic`, and and `DESKey`, respectively, though the system no longer uses the DES cipher.)

The machine stores the Smart Card Key, Smart Card Magic Number, and Data Key in a file in its internal flash memory. This file, `bs-security.cf`, resides in the same directory as `BallotStation.exe`. BallotStation encrypts the contents of the file (using AES128-CBC) with a third key called the *System Key*. However, the value of the System Key is not a secret — rather, it is the hash of the machine's serial number. The serial number is stored in the machine's registry (`HKLM\Software\Global Election Systems\AccuVote-TS4\MachineSN`), displayed in the user interface (the parameter "SN" at the bottom of every screen), and printed on the Results Report and other printouts.

As a result, any party with the ability to read data from the machine's internal flash memory can learn the values of Smart Card Key and Data Key. For example, an attacker with temporary physical access to the inside of the machine's case could exploit Issue 5.2.4 to read the contents of the `bs-security.cf` file and the registry key containing the machine serial number. The attacker could compute the System Key from the serial number then use it to decrypt the other keys.

Furthermore, malicious code running on a machine can automatically obtain the System Key, Smart Card Key, Smart Card Magic Number, and Security Key using this method. An attacker who can inject malicious code into the machine can use it to discover the keys without opening the machine's case. Alternatively, the attacker could program his malicious code to obtain the keys and use them directly to carry out another attack.

This attack may be particularly damaging because the design of the Diebold system makes it difficult to use different keys on different machines. Consequentially, all machines within a particular county most likely share the same Smart Card Key and Data Key. An attacker who can extract the keys from a single machine can therefore use them to attack all of the machines and memory cards in the county.

A malicious party who obtains these cryptographic keys can perform a number of attacks, which are described in detail later in this section. These attacks include:

- Tampering with election data files, and viewing or tampering with the electronic records of election results and system logs (Issue 5.2.6)
- Creating voter cards, supervisor cards, and election administrator smart cards (Issue 5.2.7)

[4] It is possible that an attacker with access to the inside of the machine's case could also read and write the internal flash memory using the motherboard's JTAG interface.

- Determining the order in which ballots were cast based on their serial numbers (Issue 5.2.21)

Issue 5.2.6: *Malicious code running on the machine could manipulate election databases, election resources, ballot results, and audit logs.*

Four types of data files are used in elections: election databases, election resource files, ballot results files, and audit logs. These files are stored on the removable memory cards, with backups stored in the machine's internal flash memory.

Election database files (`.edb` files) contain information about races and candidates as well as other ballot text. They are not encrypted, but they are authenticated using a kind of message authentication code (MAC). The MD5 hash of the body of the file is encrypted with the AES128-ECB algorithm using the Data Key (see Issue 5.2.5).[5] The machine requires this encrypted hash to match a value in the header of the ballot definition.

Election resource files (`.xtr` files) contain RTF strings, AccuBasic scripts, audio, images, and other data used during the election. The resources in these files are not encrypted, but each resource is individually authenticated using a MAC like the one used for the election database.

Ballot results files (`.brs` files) contain a record of the votes for each ballot cast in the election. The record of each vote is individually authenticated as in the election resource files. Each vote record is also encrypted with the AES128 algorithm[6] in CBC mode using the same Data Key.

Audit logs (`.adt` files) store a partial record of BallotStation's operation. They are authenticated and encrypted using a similar format as ballot results files, except that they use the System Key (see Issue 5.2.5) in place of the Data Key.

An adversary who obtained access to the Data Key as described in Issue 5.2.5 could carry out various attacks on these files. If the attacker had access to the memory card prior to the election, he could tamper with election files and election resource files in order to exploit other security vulnerabilities in BallotStation. Some vulnerabilities, such as Issue 5.2.15 and Issue 5.2.13 below, could allow the attacker to execute arbitrary code on the voting machine during the election. The attacker could also attempt to alter the ballots in subtle ways that would confuse voters or otherwise disrupt the election.

An adversary who knows the Data Key and has access to the memory card after the election can carry out other attacks by defeating the encryption and authentication of the ballot result files. If the attacker accesses the memory card before the votes on it are loaded into GEMS, then he can tamper with the ballot results file in order to exploit security vulnerabilities in the BallotStation terminal connected to the GEMS server (see Issue 5.2.16). Using this technique, he might be able to infect a large number of voting machines with a voting machine virus, as described in Section 3.1.

An adversary with knowledge of the Data Key and access to the memory card or voting machine after the election could defeat the encryption on the ballot result files to discover the time at which each vote was cast, potentially compromising voter privacy (see Issue 5.2.20).

An adversary with only access to the memory card or the files on the voting machine during or after the election could decrypt or tamper with audit logs that are secured with the System Key, which is not a closely guarded secret (see Issue 5.2.5). By tampering with the logs, the attacker could remove evidence of his activities. The audit logs also contain sensitive debugging information, such as stack traces recorded during errors in the software, that could help a malicious party develop further attacks.

Attacks like the ones described above could be carried out automatically by malicious code installed on the voting machine. In that case, the attacker would not need to have physical access to each machine or memory card being attacked.

[5] This non-standard MAC construction was first disclosed publicly in [33].
[6] This is consistent with Diebold's public statements (see [8]).

In the final days of this study, we learned of a recent security analysis of the AV-TSX conducted by Kiayias, et al. [24]. They found several cryptographic-related vulnerabilities in the AV-TSX that we overlooked. For instance, they found that incorrect MAC values on a record cause the record to be silently ignored but processing continues, rather than halting operation with an error message, allowing an adversary to effectively remove candidates from the ballot. Also, they found that, if a pair of candidates have names of the same length, an attacker with access to the memory card can swap the candidates by swapping their corresponding RTF-string resources, causing votes for Smith to accrue to Jones and vice versa. Due to flaws in the cryptography, these attacks do not require knowledge of the cryptographic keys. They were able to find these security holes solely by experimenting with the machine and without any access to the AV-TSX source code.

Issue 5.2.7: *The smart card authentication protocol can be broken, providing access to administrator functions and the ability to cast multiple votes.*

The AV-TSX authenticates smart cards using a two step protocol involving two secrets: the 64-bit *Smart Card Key* and the 16-bit *Smart Card Magic Number*. (Both can be set by election officials using Security Key Cards, as described in Issue 5.2.5.) When a smart card is inserted into the machine, the machine issues the ISO 7816-4 SELECT command with file ID 0x3D40 as data, then it issues the VERIFY command with the Smart Card Key as data. Conceptually, the file ID and key act like a username and password, which the voting machine uses to prove its identity to the smart card. The smart card will not allow the machine to read it unless the key the machine provides matches a key programmed into the card earlier.

If the voting machine provides the correct file ID and key, the smart card allows the machine to read the data stored on it, which vary depending on the type of card being used. Smart cards used by the AV-TSX include voter cards (which may be enabled or disabled depending on whether they have been used to cast a vote), central administrator cards, and supervisor cards. The latter two kinds of smart cards enable various supervisory functions. Part of the data stored on these smart cards is a field called the *magic number*. The voting machine requires the magic number in the smart card data to match the Smart Card Magic Number programmed into the AV-TSX machine by election officials. Thus, the magic number acts like a password that the smart card uses to authenticate itself to the voting machine.

This architecture allows a malicious voter who is given an authorized voter card and allowed to vote on an AV-TSX to steal the Smart Card Key and Smart Card Magic Number. To carry out the attack, the attacker would use a commercial programmable smart card such as a Java Card [32] to create a smart card that logs all commands sent to it. When the logging card was placed in a AV-TSX, the machine would attempt to authenticate itself to the card, and the card would record the file ID and Smart Card Key sent by the machine. Next, the attacker would place the authorized smart card in his own smart card reader and send it the file ID and Smart Card Key captured by the logging card. The authorized card would respond by granting access to its data, including the Smart Card Magic Number. This attack could plausibly be carried out by a voter on election day, since the voter will receive a legitimate voter card from poll workers and would be able to insert a logging card into the AV-TSX.[7]

With knowledge of the Smart Card Key and Smart Card Magic Number, a malicious party could carry out a variety of attacks involving counterfeit smart cards. Kohno, et al. [26] first described variations on these attacks in 2003, and they still appear to be feasible despite modifications to the smart card protocol.

For example, an attacker could use the Smart Card Key to reactivate voter cards that had already been used. Or, with slightly more information, the attacker could create new voter cards from blank Java Cards. The necessary data, which includes the election and polling place IDs, could be extracted from the voting machine by various methods, read from the

[7] Alternatively, an attacker could learn the Smart Card Key and Smart Card Magic Number using the attacks discussed in Issue 5.2.5.

election database on the removable memory card if the attacker had temporary access to it, or copied from a stolen legitimate voter card.

Furthermore, an attacker could construct a forged voter card that refuses to deactivate itself. Such a card could be used to cast multiple votes. Normally, the AV-TSX deactivates the voter card when the voter casts their ballot. This process involves rewriting the data on the card using standard ISO commands. A voter card created using a programmable Java Card could be designed to ignore those commands, or to report to the machine that the commands had succeeded while actually doing nothing. While each voter card contains a voter serial number that is supposed to be unique, the machine does not record this serial number for normal votes, so the duplicate votes would not be automatically discarded.

Finally, an attacker could use a variant of this attack to create counterfeit supervisor cards. To access supervisory functions, a poll worker needs to insert a supervisor card into the machine and key in a PIN associated with the card. Creating a forged supervisor card would require even less information than creating a forged voter card, since supervisor cards are not tied to a particular machine serial number or election ID. The PIN presents no additional challenge to the attacker. The system stores the PIN on the smart card itself, so the attacker can set it to a known value when he constructs the card.

Issue 5.2.8: *Security key cards can be forged and used to change system keys.*

Election officials use a specially formatted smart card called a *Security Key Card* to update the Data Key, Smart Card Key, and Smart Card Magic Number used by the AV-TSX. The ability to change these keys is a substantial improvement over earlier Diebold designs, which were criticized for using hardcoded security keys [26]. Standard procedures call for the keys to be changed frequently. Normally, Security Key Cards would be safeguarded by election officials and used only within the central election facility. However, a flaw in the way these cards are processed leaves them vulnerable to forgery.

Unlike other smart cards, which are authenticated with the Smart Card Key and Smart Card Magic Number set by election officials, Security Key Cards use a hardcoded key and magic number for authentication. All such smart cards apparently use the same hardcoded key, which is labeled in the source code as the "manufacturer's factory default" key. This key is the easily guessable sequence of bytes 0x01,0x02,0x03,0x04,0x05,0x06,0x07,0x08. The cards also use a hardcoded 16-bit magic number to authenticate themselves to the machine. Elections officials do not have the ability to change this key or magic number.

The key and magic number are hardcoded in the source code and are apparently the same for every AV-TSX machine in the nation. An attacker who has been given unsupervised access to an AV-TSX machine could easily extract these values from the machine. Alternatively, an attacker with prolonged physical access to a voting machine might be able to guess these values.

With these values, an attacker can use a programmable smart card to create a fake Security Key Card with arbitrary Data Key, Smart Card Key, and Smart Card Magic Number values. These values are stored on the smart card in plaintext form, so no additional secrets would be needed to construct a valid Security Key Card.

Before a Security Key Card can be loaded into the AV-TSX, the machine must be placed into supervisor mode. Normally this would require a supervisor smart card, but several attacks (see, e. g., Issue 5.2.7 and Issue 5.2.9) could allow an attacker to access supervisor mode without a legitimate supervisor card.

An attacker who is able to forge Security Key Cards and load them into the voting machines could use this attack to disrupt the election. By setting the security keys in a machine to random values, the attacker could prevent the machine from loading and saving election data (if the Data Key were changed) or from accepting supervisor and voter smart cards (if the Smart Card Key or Smart Card Magic Number were changed). At that point, the machine would be effectively non-functional: it could not be used to accept votes. A group of attackers

with forged cards could possibly reprogram the security keys on a large number of machines immediately before the election or even with voter access to the machine. Diagnosing these problems on election day might be difficult and lead to lengthy delays for voters [2].

To fix these problems, county officials would need to visit each affected machine and reprogram it with a legitimate Security Key Card. However, this would require the valid Security Key Card to be taken out of the secure central election facility, exposing it to a greater risk of theft or copying. An attacker who had momentary access to the legitimate Security Key Card could use a smart card reader and knowledge of the hardcoded "factory default" key to learn the security keys stored on the card.

See private appendix Issue 5.2.8 for more information.

Issue 5.2.9: *A local user can get to the Main Menu/System Setup menu without a smart card or key.*

Normally, the AV-TSX requires the insertion of a supervisor card or central administrator card before granting access to certain program menus. These include the "Main Menu" and the "System Setup" menu, from which the user can change hardware settings, reset the system clock, and perform other administrative functions. When the machine boots, if it detects a hardware failure in the smart card reader or the printer, it enters a fail-safe mode that allows the user to access these menus without a smart card.

The Red Team notified us that they discovered a way for a voter in the voting booth, using only a paperclip, to trick the machine into sensing a smart card reader hardware failure. This method could be used by a malicious party to access sensitive menus as part of other attacks.

Issue 5.2.10: *The protective counter is subject to tampering.*

For security purposes, BallotStation maintains a "protective counter" that is intended to reflect the total number of votes ever cast on the machine. However, as reported by Feldman, et al. [14], the counter is just a 32-bit binary value written to a unprotected file in the machine's internal flash memory (the filename is `\FFX\AccuVote-TS\system.bin`).

Attackers could change the value of the protective counter using a variety of methods. With access to the inside of the voting machine, they could alter the internal flash memory using the methods described in Issue 5.2.6. Or, by using any of malicious code injection methods described in this section, an attacker could install software on the machine to manipulate the counter. For these reasons, the protective counter is not a reliable defense against electronic attacks.

Issue 5.2.11: *SSL certificates used to authenticate to GEMS can be stolen and have an obvious password.*

Some counties use modems or other network connections to transmit unofficial election results from AV-TSX machines in polling places to the GEMS server. Optionally, these connections can be protected using the SSL protocol. In the Diebold system, the AV-TSX machines and the GEMS server have cryptographic certificates that include private and public keys, which are intended to provide strong encryption and authentication of the SSL connection.

On the AV-TSX, the certificate is stored in a file called `client.pem`, and in GEMS it is stored in a file called `server.pem`. Default certificate files ship with both systems, and it is unclear whether counties have procedures for changing them. It is likely that the same default certificates ship with most or all Diebold systems, so counties should not depend on them as a security measure. An attacker could steal the certificate file by exploiting problems like Issue 5.2.4 or by installing malicious code onto the machine.

Furthermore, while the default certificate files do use passwords to protect their private keys, both files use an obvious password—"diebold". Given that Diebold has used other obvious default passwords in the past [26], this likely would be among an attacker's first few guesses. Even without guessing, an attacker could learn this password by examining the Windows registry of a GEMS server or the application software of an AV-TSX, since the password is

stored without encryption in both places. (Access to the data on a GEMS server or an AV-TSX would usually be required to obtain the password-protected certificate file in the first place.)

Though this default password provides no meaningful security, there does not appear to be an easy mechanism for officials to change it to a secure value, since it is hardcoded into BallotStation (though not into GEMS). Thus, even if counties did replace the default SSL certificates with secure ones, they would be unable to switch to a secure password without upgrading to a new version of the Diebold software.

These problems limit the usefulness of SSL for protecting election transfers.

Issue 5.2.12: *OpenSSL is not initialized with adequate entropy.*

BallotStation uses the OpenSSL library to establish SSL connections. To operate securely, OpenSSL needs to be initialized with unpredictable data, which it later uses to generate pseudorandom numbers for use in security protocols.

When the AV-TSX boots, it initializes OpenSSL using two values — the current contents of the screen, and a deterministic function of the number of milliseconds since Windows CE started.[8] Both of these would be easy for an attacker to guess, especially if the attacker had access to machine. The contents of the screen will almost always be the same during this part of the startup process, and the time since startup should not vary by more than a small number of milliseconds. Consequently, the seed that is provided to OpenSSL's pseudorandom generator is likely to be guessable.

In a normal SSL connection, the client provides all of the secret entropy used to protect the data. Since the entropy provided by the AV-TSX is easy to guess, an attacker who can intercept the encrypted SSL traffic can use an off-line attack to decrypt it. If an attacker used this technique to obtain the ballot results file as it is uploaded to GEMS, he could try to discover how individual voters voted using the techniques discussed in Section 3.4.

More powerful attacks might also be possible. An attacker who can interpose himself between the AV-TSX and the GEMS server (for instance, by splicing into the telephone line), might be able to launch a man-in-the-middle attack in real time. With a modern computer, an attacker could guess about 1,000 possibilities for the initialization data every second. The seed provided to OpenSSL has so little entropy that an attack like this appears likely to succeed before the transmission would time out. This would give the attacker the ability to change the contents of the ballot results file in transit to GEMS and to exploit any vulnerabilities in the way GEMS processes that data.

In both these cases, attacking the SSL transmission is an alternative to obtaining access to the memory card. For further discussion of attacks on modem transmissions, see Section 4.1.8.

The GEMS application contains almost identical source code for seeding OpenSSL's pseudorandom number generator. The difference is that GEMS invokes the Microsoft Windows `CryptGenRandom()` function to generate the seed that is provided to OpenSSL. This is an excellent approach. The source code in the AV-TSX is an almost-identical copy of the source code in GEMS, except that in the AV-TSX source code the call to `CryptGenRandom()` (which is not available on Windows CE, the operating system used by the AV-TSX) is commented out and replaced by an insecure seeding process that operates as described above. In both cases, the code is immediately preceded by a comment that identifies this code as critical and warns that the OpenSSL pseudorandom number generator must be seeded properly.

Issue 5.2.13: *Multiple vulnerabilities in the AccuBasic interpreter allow arbitrary code execution.*

An earlier report revealed flaws in the AccuBasic interpreter in the AV-TSX [33]. We confirmed that those flaws remain present in the AV-TSX. This is to be expected, as we examined the same software that was examined earlier. The AV-OS contains similar

[8] The function is a kind of linear congruential pseudorandom number generator, the purpose of which might be to fool OpenSSL's entropy estimation system.

vulnerabilities, with a similar impact. See Issue 5.1.10 for more details about these flaws and their impact.

Issue 5.2.14: *Tampering with the memory card can result in code execution during voting.*

BallotStation reads the file `assure.ini` when it loads. The current election database is defined by whatever database is specified as "current" in the `assure.ini` configuration file. BallotStation assumes no line in the `assure.ini` file is longer than a fixed number of characters. If a line is longer than the relatively short buffer BallotStation has allocated for it, excess characters will be written past the end of the buffer. This would allow an attacker to crash the machine and (most likely) execute arbitrary code.

This attack is especially serious because `assure.ini` is neither encrypted nor authenticated. An attacker could modify this file without knowledge of the Data Key. An attacker only needs to be able to insert a malicious memory card into a machine, or modify a legitimate memory card that will later be inserted into a machine, to successfully compromise an AV-TSX machine.

See private appendix Issue 5.2.14 for more information.

Issue 5.2.15: *A malicious election resource file on the memory card could exploit multiple vulnerabilities to run arbitrary code.*

1. The AV-TSX displays text in the election resource file on the memory card at various stages of the election. This text is stored in language-specific RTF files. One such file contains the text used to show the current page number at the bottom of every ballot page (excluding the "cast ballot" screen). This text includes two `%d` characters which are replaced with the current page number and the total number of pages in the current ballot using `sprintf`. For example, the text "Page %d of %d" would be replaced with "Page 5 of 10" on page five of a ten-page ballot. The adversary could instead provide a string like "Page %d of %d%s%s%s%s%s%s%s%s%s%s", which would very likely crash the machine, or other strings to potentially run malicious code.

   See private appendix Issue 5.2.15 for more information.

2. The AV-TSX uses bitmap images to display information to the voter in a number of places. One such image is displayed to the voter before he inserts his card to vote. Since the image contains the language-specific message "Please Insert Your Card", it is included with other language files on GEMS and sent to the AV-TSX as part of the election database file. This vulnerability exists in this code and also in other places bitmaps are displayed to the user.

   A bitmap file begins with header structures describing its data. These headers contain a variety of information, including the size of the bitmap file and the size of the bitmap data. The AV-TSX, assuming these values will be equivalent, creates a buffer large enough to hold the bitmap data and then reads the entire bitmap *file* into that buffer. A malicious bitmap file could be constructed that would lie about its size, claiming to be smaller than it is. The AV-TSX's buffer would be too small to hold the data in the bitmap file, and a buffer overflow would occur.

   See private appendix Issue 5.2.15 for more information.

To exploit this vulnerability, an attacker would need either control of GEMS, control over the relevant RTF or bitmap files stored on GEMS, or access to the memory card. In the latter case, the attacker would need to know the Data Key. See Issue 5.2.5 for more information.

Issue 5.2.16: *Malicious election database files can cause arbitrary code execution on the AV-TSX when uploading elections to GEMS.*

When an AV-TSX uploads election information to GEMS, it calls a function that prints out an "election ticket" containing information drawn from the election database. There are three

separate vulnerabilities in this ticket-printing function that allow an attacker who is able to modify the election database to crash the machine and possibly to execute malicious code.

1. The variables containing election attributes are combined using `sprintf` into a buffer `buf` that is 512 bytes long. An attacker able to modify either of these variables could overrun `buf`.
2. The very next statement uses `buf` as an argument to an `sprintf`-style formatting function belonging to BallotStation's printer class. This formatting function contains a format string vulnerability that can be exploited by an attacker who is able to modify either election attribute.
3. Later in the ticket-printing function, a similar formatting function is called every time a file is uploaded. This formatting function contains a format string vulnerability that can be exploited by an attacker who is able to modify the a different election attribute.

See private appendix Issue 5.2.16 for more information.

An attacker able to compromise an AV-TSX machine could use this vulnerability to spread a virus to the central office AV-TSX when election results are uploaded to GEMS.

See Section 3.1 for more information on virus propagation.

Issue 5.2.17: *A buffer overflow in the handling of IP addresses might be exploitable by voters.*

A voter can potentially gain access to a screen used to set an IP address used by the AV-TSX machine. This address is copied to a fixed-size stack-allocated buffer (256 characters) and then stored in the registry using the Windows API function `RegSetValueEx`. If the IP address is longer than 256 characters, this results in a buffer overflow. The IP address is validated, but the validation function only checks that each segment of the IP address is not greater than 255 when interpreted as a 32-bit signed integer. Therefore, entering a string such as `000...0001.0.0.1` where the first segment of the IP address is more than 255 characters will crash the AV-TSX.

We do not know if it is possible to exploit this problem to execute malicious code. The input validation of the IP address entered into the system setup screen severely limits the degrees of freedom available to the attacker, so we suspect this vulnerability may not allow execution of malicious code.

To exploit this vulnerability, a malicious user needs access to the system setup screen. Issue 5.2.9 illustrates one way of obtaining this access.

See private appendix Issue 5.2.17 for more information.

Issue 5.2.18: *A malicious GEMS server can cause a crash on election download.*

When an election is downloaded from GEMS, the AV-TSX prints a label to be attached to the memory card. The fields printed on the label are treated improperly, resulting in format string vulnerabilities. An attacker with control of GEMS could place `printf` format specifiers (e. g., "%s%s%s%s%s%s%s%s%s%s...") in those fields to crash the AV-TSX machine. We do not know if it is possible to exploit this problem to execute malicious code.

See private appendix Issue 5.2.18 for more information.

Issue 5.2.19: *Ballot results files store votes in the order in which they are cast.*

The AV-TSX records votes in an encrypted ballot results file (`.brs`) stored on the removable memory card with an identical backup held in internal flash memory. The ballot results file consists of a series of records, one for each ballot cast. Each record includes a 32-bit ballot serial number, a 32-bit timestamp (in UNIX format), and a representation of the voter's selections. After each vote is cast, the machine simply appends another record to the end of the ballot results file.

This design potentially threatens the secrecy of voters' ballot selections, as first noted by Kohno, et al. [26]. An attacker with access to the removable memory card containing the ballot result files (or with access to the voting machine before the backup of the ballot results file is removed) could potentially decrypt the file to learn the exact sequence of votes cast. The attacker would need to know the Data Key used to encrypt the file, which could be obtained by exploiting the problems discussed in Issue 5.2.5, for example.

See Chapter 3 for a more complete discussion of attacks on ballot secrecy.

Issue 5.2.20: *Stored votes and VVPAT barcodes include a timestamp.*

As mentioned in Issue 5.2.19, ballot results files include a 32-bit UNIX-style timestamp with the record of each cast ballot. This information could allow an attacker who can access the ballot results file and who knows the Data Key to compromise ballot secrecy. For example, if the time when each voter checks in is recorded in the poll log book, an attacker with access to the log book could correlate this data with the timestamps to determine how voters voted. Alternatively, observers in the polling place could note the time when target voters cast their votes and find the corresponding vote records in the ballot results file.

The machine also encodes the time when each vote was cast as part of the barcode that is optionally printed on each VVPAT record. If election officials have enabled printing of barcodes, this design provides another opportunity for attackers with insider access to match votes with voters' identities. Procedures for handling and auditing the ballots could mitigate or exacerbate this vulnerability. In particular, if the ballot barcodes are scanned as part of the audit process, access to the resulting data must be carefully safeguarded. Fortunately, GEMS provides a configuration option to disable printing of barcodes on VVPAT records, in which case the timestamp will not be included on VVPAT records but will still be included in the ballot results file records.

Issue 5.2.21: *Ballot serial numbers are chosen using an insecure method, which may allow attackers to discover the order in which ballots were cast.*

The AV-TSX assigns a serial number to each ballot cast. The serial number is recorded as part of the election results file, optionally printed on the paper ballots in barcode form, and displayed in various parts of the AV-TSX user interface. To protect voter privacy, the AV-TSX attempts to hide the order in which ballots are cast by assigning ballot serial numbers using a cryptographic pseudorandom function.

Kohno, et al. [26], in their study of an earlier version of Diebold's software, discovered that the mechanism used to assign these pseudorandom serial numbers was insecure and could allow an attacker to recover the true order of the votes. Diebold has since updated the software to use a different method, but attacks are still possible in spite of the changes.

Today, BallotStation assigns serial numbers by (essentially) running a custom 20-bit block cipher in counter mode. The block cipher is based on a Feistel network, using AES as the round function. (Comments in the source code refer to section 14.10 of Bruce Schneier's *Applied Cryptography*.) The network consists of 8 rounds, where the ith round computes, for a 10-bit half x_i, $c = \text{AES}(x_i)$ and extracts the 10-bit substring beginning at bit $16 \cdot i$. Serial numbers are encrypting a 20-bit counter using this block cipher. The software requires serial numbers to be no greater than 1,000,000, and it skips over ones that do not satisfy this constraint.

For this scheme to be secure, the encryption key for the AES cipher must remain secret. BallotStation generates this key by taking a value called the *BRS Signature* and encrypting it with the Data Key. It generates the BRS Signature when the ballot results file is created by taking the MD5 hash of several values from the results file header, such as the machine serial number, plus the system's tick count (the number of milliseconds since the machine booted).

Most of the values from the election header are printed in election logs or displayed in the voting machine user interface, so an attacker could launch a brute force attack to guess the

system tick count. However, an easier attack may be possible, since BallotStation uses the BRS signature value as part of the filename for the ballot results files. This means that an attacker who could obtain the Data Key (see Issue 5.2.5) and determine the name of the ballot results file could quickly calculate the list of ballot serial number in the order in which they were cast. With this information, an election insider might be able to match the VVPAT records to individual voters, or an attacker with a copy of the ballot results file could determine the order in which all votes were cast, even if Issue 5.2.19 and Issue 5.2.20 were fixed.

Issue 5.2.22: *Files on the voting machine are not securely erased when they are deleted.*

BallotStation deletes files from the memory card or the machine's internal flash memory at various times during the election process. For instance, it allows election officials to delete the backup copies of old ballot results and audit logs that are stored on the machine. Deleting old these files is important for safeguarding the secrecy of the ballot, since, as described above, the contents of the files may reveal how voters voted.

However, when BallotStation delete files, it does so using the standard Windows `DeleteFile()` API. This function removes the file from directory listings, but it does not securely erase the contents from the memory card. Even after being deleted, the data will remain in the memory card until it are overwritten with other data, which may take multiple election cycles. An attacker with access to the memory card, unsupervised physical access to the machine, or the ability to run malicious software on the voting machine might be able to recover the contents of these files even after BallotStation deletes them.

Issue 5.2.23: *Logic errors may create a vulnerability when displaying bootloader bitmap images.*

When the machine boots, it uses incorrect code to display a bitmap image. Since this code only loads images from the bootloader memory, an attacker would need to be able to modify the bootloader (for instance, using Issue 5.2.1) in order to exploit this issue. However, an attacker with the ability to modify the bootloader could simply insert malicious code directly, without having to exploit the bitmap display error. Therefore, the impact of this vulnerability is low, but we describe it in detail as an example of code quality problems in the bootloader and because it underscores the challenges of thorough code review and testing.

The following is a short except of the bootloader bitmap display code:

graphics.c:253

```
253  void GlibPutPixel(UINT xx, UINT yy, Pixel_t Color)
254  {
255      // Check for library not initialized or (x,y) out of range
256      if(FrameBuffer != FALSE || (xx < USER_X) || (yy < USER_Y))
257      {
258          // Compute the frame buffer offset and write the pixel
259          FrameBuffer[FB_OFFSET(xx,yy)] = Color;
260      }
261  }
```

The `GlibPutPixel` function is used to write pixels from a bitmap into a buffer. Line 256 was intended to check that the data being written lies within the boundaries of the buffer. However, the programmer mistakenly employed logical "or" operations where logical "and" operations are required. The correct code differs in a subtle but crucial way:

```
if(FrameBuffer != FALSE && (xx < USER_X) && (yy < USER_Y))
```

The result is that the boundary checks have no effect. As long as the `FrameBuffer` condition holds, the `if` statement will succeed. As constructed, `GlibPutPixel` would allow a specially crafted bitmap file (embedded into the bootloader by an attacker) to overwrite portions of memory, possibly leading to the execution of malicious code.

Issue 5.2.24: *AV-TSX startup code contains blatant errors.*

startup.cpp:287

```
287   TCHAR name;
288   _stprintf(&name, _T(''\\Storage Card\\%s''), findData.cFileName);
289   Install(&name, hInstance);
```

Here, `name` is not a character array but a single character in memory. The `_stprintf` function expects its first parameter to be a character array, so the programmer had to use the & operator to get the address of `name`, rather than its value. The result is an obvious buffer overflow. A string that includes the filename, which could be under an attacker's control, gets copied over whatever data resides in the memory region following `name`.

That this code works at all seems purely accidental. Memory corruption occurs even when legitimate .ins files are used. An attacker who included a file with a long name or a name containing particular characters might be able to crash the program or, possibly, execute malicious code.

This bug sheds light on the vendor's software engineering practices, because it is a very unusual error for an experienced C++ programmer to make. Characters and character arrays are very different constructs in C++. Students using the language for the first time might confuse the two, but experienced programmers who understand basic concepts like pointers would be unlikely to confuse them. The probability that an experienced C++ programmer would make such a mistake or overlook it during even a cursory review of the code is exceptionally low. This suggests to us that after this code was written it was not reviewed by any other engineers at Diebold.

5.3 GEMS

In analyzing GEMS, we focused on several interfaces that serve as points of entry for external data. These interfaces allow for:

- Communication with AccuVote-TSX machines via TCP connections
- Communication with AccuVote-OS machines via raw byte streams
- User input via the graphical user interface
- Interaction with the database

There are other interfaces, such as data importation, that we were unable to examine within the limited time of our review. Therefore, other problems may exist.

Many of the most dangerous vulnerabilities that we found in GEMS relate to its use of and interactions with the database. In general, data from the database is fully trusted by GEMS. The developers apparently made an implicit assumption that malicious parties would be unable to subvert GEMS itself or to modify the database outside of GEMS. On the GEMS server that the Red Team received from Diebold, such trust does not appear to be warranted. According to the Red Team, Diebold representatives indicated that the configuration of the GEMS server that we received matches what a county typically would receive [3].

We identified the following specific issues with GEMS:

Issue 5.3.1: *GEMS uses the Microsoft Jet data layer.*

Essentially, a data layer is the component that directly modifies the underlying data store of a database. A data layer may be combined with an application layer, such as Microsoft Access, that allows users to more easily interact with that data layer. In practice, the term "Access" typically refers to the combination of the Access application layer with the Jet data layer [7].

Microsoft offers two popular data layers, Jet and SQL Server, to meet the varied needs of diverse users and organizations. According to Microsoft documentation "The strength of Access [Access/Jet] is its ease of use, rapid application development environment, and simplistic distribution... The strength of SQL Server is its more robust data integrity,

scalability, security, and manageability." [7] While Jet may be appropriate and even desirable in a wide variety of cases, other options seem more appropriate for a critical election scenario. For example, Microsoft recommended against the use of Jet in Cuyahoga County, Ohio due to the large scale of the county's election [12]. Ryan and Hoke [30] discuss additional issues with the use of Jet.

Even if GEMS used Jet with extreme caution and implemented any additional necessary features above the data layer, Diebold's choice of Jet is questionable when alternative solutions exist that seem more appropriate. Barring a compelling reason, Diebold should switch to SQL Server or another appropriate alternative.

Issue 5.3.2: *Anyone with access to the GEMS server's local disk can modify the GEMS database.*

Anyone with access to the database files could use Microsoft Access to modify them, even while GEMS is running. GEMS does not lock the entire database while it runs, so modification of any value in the database seems possible at nearly any time. This allows malicious insiders to bypass the access control and other restrictions in GEMS.

We have successfully modified a database using a hex editor, so Access is not even necessary.[9] Given access to the disk, a malicious user could create custom software to carry out elaborate attacks against the database automatically (e. g., see Issue 5.3.7). Using exploits discussed in the Red Team report, an attacker may even be able to hack into the GEMS server over the local area network and modify the database [3].

We have tested modification of election results, audit logs, candidate names, user names, user passwords (in combination with Issue 5.3.8), user access levels, and a variety of other data using programs other than GEMS.

Issue 5.3.3: *GEMS trusts the graphical user interface (GUI) to safeguard data and enforce security constraints.*

In a number of cases, GEMS relies on the state of widgets to enforce when and how users may modify data in the underlying database. For example, if a user without administrator access attempts to edit her user options, she will see a window in which the "Administrator" checkbox is blank and disabled. The disabled checkbox is the only safeguard that prevents the user from giving herself administrator access. Using a small, freely available program, we successfully enabled disabled widgets, and GEMS automatically updated the database to match those changes.

Were users without administrator access able to install similar code on a GEMS server (for example, by installing a CD), they could give themselves administrator access. Through a series of steps, a malicious user without administrator access would even be able to delete the primary administrator account.

The ability to subvert GUI-enforced controls in GEMS allows for a variety of other attacks. For example, vote totals are reported based on numeric identifiers that map to candidates. An attacker could change the mappings even after the election has begun, meaning that the GEMS server would have different mappings than the voting machines. While we have not tested this attack, we believe that it would cause vote totals to be reported for incorrect candidates (the Red Team has tested a similar attack for us — see Issue 5.3.5).[10]

Since this attack subverts the GEMS application's own user interface to modify election data, it is simpler than other ways of tampering with data in memory, and it could be carried out by an attacker with relatively little technical skill.

Issue 5.3.4: *Procedures described in Diebold system documentation place too much trust in third-party transcription and translation services.*

[9] For our tests, we needed to close GEMS prior to modification, but it may be possible to modify the database while GEMS is running using programs other than Access.

[10] Changing identifiers can change the order in which candidates appear on results reports, but election officials would need to know that the order is incorrect to catch this.

GEMS and the AV-TSX support multiple languages through language-specific text (RTF) files. Chapters 5.4.2.2 and 5.4.2.3 in the GEMS Election Administrator's Guide [10] describe the process of translating ballot information from English into a foreign language using a third-party translation agency. These processes appear to place inappropriate trust in the translation agency.

Two processes are documented:

1. (5.4.2.3) Send the entire GEMS database to the translation agency. The translation agency modifies the database directly using its own copy of GEMS. The agency then returns the modified GEMS database, which is considered to be the official election database.

   If the translation agency is malicious, it could embed malicious data into the database that will compromise GEMS (Issue 5.3.7), all AV-TSXs, and all AV-OSs in the county. The Election Administrator's Guide does state that this option "requires that the translation agency. . .not compromise the integrity of the database and the election." The guide does not note, however, that a malicious translation agency can subvert all GEMS and AV-TSX software in the county. This level of trust in translation agencies is dangerous.

2. (5.4.2.2) Send the RTF files to a translation agency by invoking the GEMS "Export RTF" function. The translation agency edits the RTF files and returns files containing translations. Officials can use the GEMS "Import RTF" function to load the translated files into GEMS.

   The Election Administrator's Guide says nothing about trust in the translation agency when following this process. This could lead a security-conscious election administrator to believe that sending only RTF files to a translation agency would result in no security problems. Unfortunately, a malicious translation agency with access to only the RTF files could crash or possibly run malicious code on the AV-TSXs used for voting (see Issue 5.2.15).

Similar issues apply to the use of third-party audio recording agencies.

Issue 5.3.5: *Race and candidate labels may be changed after GEMS has been "set-for-election."*

GEMS allows election officials to change race and candidate labels after the election data has been transferred to the voting machines. Comments in the code indicate that this feature is desired to allow for correction of spelling errors, but the comments acknowledge that this raises a "big security issue." Results from voting machines are reported by numbers mapped to candidates and race names, not by the names themselves. Thus, swapping the names of two candidates or races will cause votes to be attributed to the wrong candidates or races in GEMS.

An attacker would have to solve several challenges to exploit this weakness. The attack would need to occur on the master GEMS server — otherwise, the candidate numbers would map to the correct names on the machine that compiles results. Also, party labels cannot be swapped (we ignore the possibility of changing party affiliations using Issue 5.3.3). Since swapping names could therefore cause mismatched party affiliations, swapping two candidate names alone might arouse suspicion if results include both candidate name and party. In addition, swapping yes/no answers for ballot issues would result in a no/yes ordering that might look suspicious.

Some of these challenges do not apply in certain cases, such as party primaries, and others can be overcome. For example, by changing both candidate and race names, an adversary could cause the results:

```
Mayor
   Candidate A (Rep)  550
   Candidate B (Dem)  450
```

```
District Attorney
   Candidate C (Rep)  400
   Candidate D (Dem)  600
```

to instead be reported as:

```
District Attorney
   Candidate C (Rep)  550
   Candidate D (Dem)  450

Mayor
   Candidate A (Rep)  400
   Candidate B (Dem)  600
```

Candidate A should have won the mayoral race, and Candidate D should have won the district attorney race. Based on these values, however, Candidate B would win the mayoral race, and Candidate C would win the district attorney race.

These attacks can be easily detected during the official canvass by comparing summary tapes printed at the polling place against the official results produced by GEMS.

A better solution for spelling errors might be to allow officials to append notes to reports rather than allowing officials to change actual labels. This would also keep officials from using this flaw to accidentally or purposely cover up spelling or labeling errors that could have confused voters during the election.

Although this vulnerability is unfortunate, honest documentation of potentially dangerous design decisions and the corresponding rationale is tremendously useful for both reviewers and future developers and should be encouraged. We emphasize that the problem here is not the presence of the source code comment quoted earlier, and the solution is not to delete that comment from the code. Difficult design decisions are sometimes necessary and always warrant justification. The problem here is that, even when potential issues were documented, quality assurance processes failed to produce less dangerous alternatives.

Issue 5.3.6: *GEMS fails to filter some user input before using it in SQL statements.*

When requesting log-in credentials from a user, GEMS places the username input string directly into a SQL statement without filtering that string. This would allow an attacker to conduct a minor SQL injection attack against the database. A carefully crafted username string would allow a malicious user to gain additional database information. For example, assume that GEMS executes the query:

```
SELECT password FROM users WHERE username='username';
```

(where *username* is the user input) and compares the user's password (from the database) to the typed-in password to determine whether the user should receive access.[11] If the user already has an account, the user is able to ask true/false questions about the contents of any table in the database. For example, suppose that a malicious user `mallory` wishes to learn whether `alice` has administrator access to GEMS. Rather than entering `mallory` as her username, she could enter:

```
mallory' AND (SELECT COUNT(*) FROM users WHERE
username='alice' AND access='admin')=1 AND
'1'='1
```

This would cause GEMS to execute the SQL statement:

[11] This is a simplified version of the GEMS login process.

```
SELECT password FROM users WHERE username='
  mallory' AND (SELECT COUNT(*) FROM users WHERE
  username='alice' AND access='admin')=1 AND
  '1'='1';
```

If `mallory` enters her correct password and can log in, she learns that `alice` must be a user with administrator access. Otherwise, she learns that `alice` does not have administrator access.

While this particular vulnerability does not appear to allow modification, addition, or removal of data, this is only due to the apparent constraints of Jet. Based on our tests, Jet does not allow statements that can modify the database to appear inside `SELECT` queries, and it does not allow multiple SQL statements to be sent in one SQL query. However, relying on these limitations of the database engine is bad practice, as they may change in future versions.

Issue 5.3.7: *In several cases, GEMS trusts data from the database not to be malformed.*

Multiple buffer overflows are possible if an adversary is able to modify data in the GEMS database. Data including text preferences and district information is trusted to be less than a given length. All identified buffer overflows appear to impact data on the stack, meaning that a straightforward "stack smashing" attack would be possible.

In one case, GEMS copies data pertaining to election districts from the database into a `CString`, resulting in a `CString` that may be arbitrarily long.[12] GEMS then creates a constant-sized buffer on the stack and copies a slightly modified version of the `CString` character-by-character into the buffer — and, if the string is too long, beyond the end of the buffer. In our tests, we are able to crash GEMS by changing text in the relevant field of the database. We expect that an adversary with access to the database could use the vulnerability to execute arbitrary code on the GEMS server.

In another case, GEMS copies a table entry relating to various formatting preferences from the database to a `CString`. It then parses that string, copying an arbitrary-length substring pertaining to text formatting into a constant-sized buffer. More than one problematic path to the flawed code exists, but we believe that the buffer is on the stack in all known cases. We were able to crash GEMS by modifying any one of multiple entries in the database and believe that an adversary could exploit the flaw to execute arbitrary code.

See private appendix Issue 5.3.7 for more information.

Issue 5.3.8: *Attackers can create a valid "encrypted" password from any desired user password, without needing to know any cryptographic keys.*

To encrypt user passwords for storage in the database, GEMS calls the OpenSSL function `DES_crypt` with the plaintext password and a two character salt as parameters. To generate the salt, the program relies on the `rand` function, seeded by the present time. `DES_crypt` uses the salt to introduce disorder in the DES encryption algorithm, and it uses content from the first eight characters of the password as a key. The algorithm uses the key to encrypt a static string, and `DES_crypt` returns a string containing the two-character salt followed by the algorithm result [34, 5]. GEMS writes the returned string to the database. While GEMS calls this "encryption", it would be more accurate to describe this as hashing the password and salt using a deterministic hash function.

This process is problematic for several reasons. First, any user with access to the database can choose a secret password and salt, create an encrypted password, and replace the password in the database with the new value known. If this new password is known only to the user, this could be used to take control of other users' GEMS accounts.

[12] While Access/Jet has length limits depending on the field type, the attacker can simply change the field type to permit arbitrary-length values. This attack does not rely on changing the field type, however, as the present field type allows sufficiently long values.

Second, a user with a valid GEMS account but without access to the database could use Issue 5.3.6 to rapidly learn the salt of another user. Even a naïve approach of guessing the first character then the second character would require 128 or fewer tries, and a user could cut this to twelve or fewer using binary-search techniques. The expected number of guesses grows linearly with the length of the salt. The same techniques could be used to learn the other user's encrypted password. Given the salt and encrypted password, a malicious user would have everything necessary to guess the corresponding password without additional interaction with GEMS or modification of the database. If the password is relatively weak, this process may be almost trivial with an off-line dictionary attack [25]. If not, the process would be far more difficult: an attacker could be reduced to brute-force guessing eight random characters. Nevertheless, such attacks are not unprecedented [27], and an adversary able to obtain special-purpose hardware could conceivably learn the password in time for an election within months or fewer after learning the salt.[13] Changing passwords regularly or using both longer passwords and an encryption function utilizing longer keys might help mitigate any potential for password cracking. That said, the impact of this weakness appears to be relatively minor, compared to other issues described here (e. g., Issue 5.3.2, Issue 5.3.7).

Issue 5.3.9: *In several cases where GEMS converts signed integer values to strings, GEMS writes them into buffers that are too short.*

In several cases, GEMS copies signed integer values into ten-byte character buffers. In addition to the ten digits that a 32-bit signed integer may contain, its string representation requires a terminating character and may require a negative sign, making a buffer of twelve bytes necessary. While an attack could potentially overwrite the buffer by up to two bytes, the executable is compiled with DWORD alignment and, consequently, the data following the buffer will start at least two bytes past its end in each of these cases. This means that the flaw probably is not exploitable. Nevertheless, as pieces of code tend to migrate elsewhere in a project, these errors should be corrected. In addition, this code may eventually be used in an application without DWORD alignment.

We have not confirmed that making the integer value negative or large enough to cause a buffer overflow is possible in every one of these cases but we still consider this code dangerous.

See private appendix Issue 5.3.9 for more information.

[13] The COPACOBANA system, utilizing approximately $10,000 of special-purpose hardware, reportedly is capable of cracking DES in under nine days on average [27].

CHAPTER 6

Procedural Safeguards and their Limitations

In this chapter, we discuss the extent to which changes in election procedures can compensate for the security shortcomings of the Diebold software. Although we believe that some of the procedures that we describe below may help reduce the risk posed by these vulnerabilities, we stop short of endorsing any of them because we are not confident that anything short of a redesign of the Diebold system can provide an acceptable level of security.

6.1 Logic and Accuracy Testing

Logic and accuracy testing provides little defense against software-based attacks. Malicious software running on the machines can detect whether officials are performing logic and accuracy tests and can force the machine to behave normally until the testing completes.

6.2 Commercial Virus Scanners

Commercially available virus scanners provide little defense against the kinds of attacks described in this report. They normally are only able to recognize PC viruses that have been observed in the wild on many computers. However, they cannot detect new attacks never seen before, and they are not designed to detect virally propagating malicious code that targets voting equipment and voting software.

6.3 Stricter Chain of Custody Measures

We are not optimistic that stricter chain-of-custody controls will prove effective in addressing the vulnerabilities identified in this report. We were not able to identify any realistic procedures that would ensure that voting equipment and memory cards remain under two-person control at all times. Leaving voting machines unattended overnight in a polling place breaks the chain of custody and creates an opportunity for an attacker to tamper with the machines. Sending voting equipment home with the chief poll worker allows that person unsupervised access to the equipment; since in many counties essentially any registered voter who volunteers can become a poll worker, it is difficult to prevent an attacker from becoming a poll worker. Since it might take only one compromised machine to spread a virus to all the county's voting machines, the prospects for devising chain-of-custody rules that will meet the necessary level of perfection in practice seem dim.

6.4 Tamper-Evident Seals

We do not expect that tamper-evident seals will be effective at detecting tampering with voting equipment while it has been left unattended. First, the Diebold polling place equipment does not appear to have been designed to meet this threat model. We understand that the Red Team has identified several ways that a voter might be able to tamper with an AV-TSX machine while in the process of voting [3]. Second, most, if not all, tamper-evident seals have known vulnerabilities that could allow an attacker to break them and then replace them or restore them to a condition where the tampering is unlikely to be detected [21]. Third, it is challenging to devise protocols that make it likely that poll workers will detect and respond appropriately to tampering; few poll workers have prior training as a seal inspector, and it is not practical to provide the kind of training that would be needed in the already-rushed training that poll workers receive. Fourth, the false alarm rate (where seals are broken or unverifiable for innocuous, ordinary reasons) is so high that election workers may become inured to these issues; it may be difficult to ensure that broken seals are consistently taken seriously enough. Since it only takes one compromised machine to infect the entire county's voting machines, we do not believe that tamper-evident seals can prevent introduction of virally propagating malicious code. See Section A.5 for a further discussion of tamper-evident seals.

6.5 Forensics

Forensics performed after election day may be helpful to determine the cause and nature of attacks. However, procedures to govern forensic analysis should be in place before any problems are detected. Viruses and other malicious software could be designed to remove traces of their activities from the voting machines at the end of the election, so workers need to collect and preserve evidence even before they suspect an attack. Ideally, some number of voting machines and memory cards should be randomly selected and set aside, unused, so that any attack software present will be preserved for analysis.

6.6 Parallel Testing

Parallel testing is another partial mitigation to consider. Parallel testing involves selecting a random sample of DRE machines, taking them aside, and running a mock election on election day using the equipment. By preparing a known voting slate, one can compare the results from those machines against the inputs that mock voters entered. Typically, parallel tests are videotaped so that it is possible to go back and review any discrepancies. Parallel tests are one way to detect bugs or malice in DRE software, if the faulty software is widespread enough that the random sample is likely to pick at least one DRE that exhibits incorrect behavior.

The reliability of parallel testing at detecting malicious code appears to be open to debate. The effectiveness of parallel testing is heavily dependent upon the details of how the testing is done. If malicious software can distinguish when it is being tested from normal operations, for instance by looking for mistakes that inexperienced voters would make but officials performing tests would not, then the malicious software can evade detection by behaving correctly when it is under test.

Ultimately, parallel testing becomes an arms race between attack designers and officials who plan realistic parallel tests. The defenders attempt to design testing procedures that mimic real elections as closely as possible, while we must assume the attackers will try to design methods to detect when they are being tested. It is not clear who has the advantage in this race. The problem with this kind of arms race is that it is difficult to know who is winning. Thus, there is a risk that an attacker might develop a secret way to defeat parallel testing, leaving the defenders with a false sense of security about election integrity.

Another way to thwart parallel testing would be to use a secret knock (a series of inputs known only to the attacker that would be unlikely to happen by chance) to control activation of the vote-stealing code [4]. A secret knock could be used to activate the virus, though this would require the

virus author to have conspirators who could access each of the voting machines where votes would be stolen. Alternatively, a secret knock might serve to deactivate the vote-stealing code, though this would require the help of an insider in the parallel testing process.

Parallel testing only defends against malicious software on the AV-TSX DREs. It does not defend against malicious software at county headquarters, such as malicious software on GEMS or on the central-office AV-TSX/AV-OS units.

Parallel testing is more effective at detecting attacks than at preventing them from disrupting the election. Suppose testing reveals that a small number of votes are recorded for the wrong candidate. If the test is conducted on or close to election day, there may not be enough time to determine the cause. As described above, it may be difficult or impossible to determine the correct vote totals if attack code is running on the machines on election day. Denial of service attacks present an even greater challenge. Officials have few recovery options if they discover shortly before the election that the machines will fail the next time they are used, and parallel testing on election day provides no advance warning of such a failure.

Finally, unless parallel testing is performed on a very large number of machines, it will have a low probability of uncovering attacks that are directed only at specific precincts or election conditions. Other mitigation strategies must be applied to control these risks.

Despite these limitations, parallel testing may still have value at detecting viral attacks and human factors attacks on the VVPAT, like those discussed in Section 3.3. We leave it to others to analyze the cost-effectiveness and appropriateness of parallel testing.

6.7 Voter-Verifiable Paper Records

One of the critical security mechanisms in the Diebold voting system is the voter-verifiable paper trail, namely, the paper ballots for the AV-OS and the VVPAT records for the AV-TSX. The idea is that, in case an attacker manages to replace the certified software on the AV-OS or AV-TSX with malicious software, the paper trail will provide a way to detect misbehavior by the malicious software. Any strategy to mitigate the Diebold system's technical problems must take into account the limitations of the paper trail system. Therefore, we discuss the constraints these limitations would impose on any solution.

Voter-verifiable paper records (paper ballots and VVPATs) are perhaps the best defense against vote-stealing attacks; however, as discussed in Chapter 3, they may not be adequate to detect and recover from attacks that change only a small number of votes. The design of the paper audit trail greatly influences its effectiveness. Voters should be strongly encouraged to review the contents of the VVPAT record and to report any discrepancies to poll workers. Discrepancies should be logged and reported to election officials and centrally tracked on election day to monitor for signs of a widespread problem.

VVPATs provide little defense against most kinds of denial-of-service attacks, since the machines cannot print VVPAT records if they are not operational. Attackers may also target the VVPAT directly, for instance, by programming the machine to exhaust the supply of paper.

6.8 Ballot Secrecy Protections

The ballot secrecy problems identified in this report are difficult to mitigate. After-the-fact procedural controls seem inherently inferior to technological measures that randomize the electronic records at the time the vote is cast. We recommend that the AV-TSX software be fixed to ensure that the electronic records retain no trace of information that might reveal voter identity and to ensure that the electronic cast vote records are independent of the order in which voters voted.

Until the software can be fixed, there may be no fully satisfactory solution, but we can identify several stop-gap steps that election officials could consider adopting if they must use the AV-TSX:

- Do not record voter names in the sign-in roster sheet in the order that voters sign in. For instance, one might use roster sheets that have voter names pre-printed in alphabetical order.

- Do not use e-pollbooks that record or transmit any information about the order in which voters signed in.
- Consider introducing procedural mechanisms to ensure that county staff who are present at a polling place are not given access to the electronic or VVPAT records for that polling place.
- Limit the number of individuals with access to the GEMS network to the minimum necessary, and ensure that they can be trusted. Do not give temporary workers access to GEMS, the network that GEMS is connected to, or other devices that are connected to GEMS.
- Limit or prohibit access to the raw ballot results files stored on the memory cards.
- Use the Key Card Tool to change the cryptographic key on every AV-TSX machine in the county to a secret, county-specific, unguessable key, and establish strict two-person control over all AV-TSX memory cards that contain electronic records of voted ballots. These measures make it harder for a malicious poll worker, who is entrusted with transporting the memory card back to county headquarters, to make a copy of the electronic results files for later analysis.
- Offer every voter the opportunity to vote on a paper ballot. Optically scanned paper ballots are not subject to the ballot secrecy risks in the AV-TSX. Unfortunately, if the county does not have an AV-OS scanner in every polling place, voters who vote on paper ballots do not receive the benefit of overvote notification and thus may suffer from a higher rate of lost votes. Counties in this position should adopt procedures to minimize the rate of lost votes, such as screening all centrally counted paper ballots cast at the polls for overvotes or marginal marks and manually examining those ballots for voter intent.

These mitigations are incomplete. Even if all of them were adopted, they still would not suffice to completely address the ballot secrecy shortcomings of the AV-TSX.

The risks to ballot secrecy only apply to voters who vote on the AV-TSX. Therefore, voters who use paper ballots are not subject to these risks. The impact of the AV-TSX ballot secrecy issues will be proportional to the number of voters who use the AV-TSX; counties who use a hybrid model and encourage most voters to use paper ballots will be affected less than counties where all voters vote on the AV-TSX.

6.9 Minimizing Use of Modems and Shared Networks

Modems Modems pose a risk to election integrity. The risk is that someone may be able to dial into the modems and compromise GEMS. If someone is able to do that, they may be able to introduce virally propagating malicious code onto the server, which will then be able to infect all the voting machines in the county in the next election. We are concerned that it would be easy for software flaws or misconfiguration associated with modems to make it possible for someone to mount such an attack. See Section 4.1.8 for further analysis.

The safest countermeasure is probably to simply avoid any use of modems. Under this approach, modems would not be used for any purpose, not even to communicate unofficial results on election night, and GEMS would never be connected to any modem or to the public telecommunication system (or to any machine or network that is directly or indirectly connected to such) at any time. The advantage of this approach is that it eliminates the possibility of a physically remote attacker dialing in and hacking the voting system. The disadvantage of this approach is that it may slow down the reporting of election results, particularly for geographically distributed counties.

Regional processing The use of regional processing also poses similar risks, because it inherently involves connecting GEMS to modems or shared networks.

Shared networks pose a risk that is analogous to that posed by modems. The degree of risk may depend upon many factors, such as how many people have authorized access to the network,

how easy it may be to gain unauthorized access to the network, and what else that network is used for. At one extreme, connecting any component to the public Internet is very dangerous, because it creates the opportunity for anyone anywhere in the country to attack the system. At the other extreme, a physically secured point-to-point communication link may pose little risk. County intranets and other shared networks may fall somewhere between these extremes and the risk they pose depends upon the specific circumstances.

We mention several potential responses to this risk, in order of decreasing security:

1. The most secure countermeasure is to avoid any use of modems or other shared networks and to avoid using regional processing. This eliminates the possibility for a physically remote attacker to dial in and breach the security of GEMS.

2. A closely related variant is to use regional processing but avoid any use of modems or networks. Instead, unofficial results could be transported from regional return centers to county headquarters on write-once media (e. g., CD-R or DVD-R).

3. Another possibility is to use a physically secured point-to-point communication link, such as a dedicated T1 line leased from the phone company. This may partially reduce but not eliminate the risk. There remains the risk of configuration errors, attacks by phone company insiders, breaches of security within the phone company, and breaches of physical security at either endpoint.

4. A third approach is to use a commercial virtual private network (VPN) product to emulate a point-to-point link. This carries the risk of configuration errors as well as the risk of software flaws or cryptographic weaknesses in the VPN product. Unfortunately, configuring VPN products securely can be tricky, and determining whether a VPN product has software flaws is difficult. Therefore, this option may not be as safe as a dedicated point-to-point link.

5. Using modems is the riskiest approach. Because of the risk of configuration errors and software flaws, we would not recommend this option even to expert system administrators.

Counties may wish to re-examine the security risks associated with regional processing and assess whether the benefits outweigh the risks.

6.10 A Segregated Dual-GEMS Architecture

Another potential approach that is worth investigating involves deploying two separate GEMS installations at county headquarters, a permanent GEMS and a sacrificial GEMS. The permanent GEMS installation would be used for laying out the ballot, defining the election, and writing to memory cards before the election. The sacrificial GEMS installation would be used for reading memory cards, accumulating and tabulating results, and producing reports. The latter installation can be reformatted after the election and is never used to write memory cards, so if it is infected by a virus, at least the virus will not be able to spread to every other voting machine in the county.

This architecture is motivated by the observation that the key step in the propagation of the virus of Section 3.1 is when an infected central-office unit is used to write many memory cards destined for the field, infecting all of them. This step is what causes the virus to spread so rapidly. If we can ensure that no infected central-office machine unit is ever used to write memory cards, then we can prevent the rapid viral spread of Section 3.1.

The approach In more detail, we would have two entirely separate, isolated installations of GEMS. Each would be a complete installation of GEMS and accompanying equipment, complete with its own Ethernet network, port server, and central-office AV-TSX and AV-OS systems. The

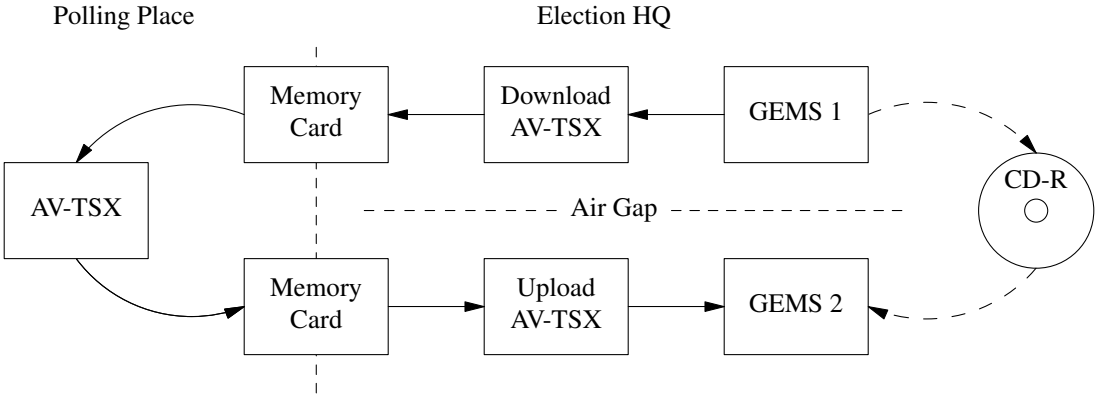


Figure 6.1: Adopting a **segregated dual-GEMS architecture** would help protect against certain kinds of viruses. Officials use one GEMS server and set of central-office voting machines to create memory cards before the election. They use a second, physically separate GEMS server and set of voting machines after the election to tabulate results. The GEMS database is transferred from the first GEMS server to the second GEMS server using a write-once medium, such as a CD-R.

two systems would be carefully segregated and air-gapped[1] to ensure that there are no cross-connections. The sacrificial GEMS installation would be treated as presumed-to-be-infected, so any machine or equipment that is ever connected to the sacrificial GEMS system must never again be connected to the permanent GEMS installation. Strict procedural controls must be applied to ensure that any media that has been connected to the sacrificial GEMS installation is securely erased or reformatted before being used with the permanent GEMS installation.

Before the election, system administrators would reformat and reinstall all the machines and software on the sacrificial GEMS installation, to bring up a clean sacrificial installation. County staff would use the permanent GEMS installation to lay out the ballot, define the election, and program all of the AV-OS and AV-TSX memory cards. Then county staff would write a backup of the GEMS database from the permanent GEMS installation onto write-once media (e. g., CD-R or DVD-R), carry the media by hand to the sacrificial GEMS installation, and install that GEMS database onto the sacrificial GEMS. After this point, the permanent GEMS installation would not be used for the remainder of the election.

On election night, as memory cards or other equipment are returned from the field, they would be taken to the sacrificial GEMS installation (*not* the permanent GEMS installation). Memory cards would be read using the central-office AV-TSX and AV-OS units that are part of the sacrificial installation. The sacrificial GEMS would be used to accumulate and tabulate election results, produce reports, and calculate the official election results.

Finally, after the election is over, all memory cards would be erased and reformatted using a separate laptop (not connected to either GEMS installation) that is used only for this purpose. This ensures that if the memory cards were carrying data infected with a virus, they have been returned to a clean uninfected state. Some mechanism would have to be devised to securely reformat the AV-OS memory cards.

System administrators could optionally reformat all devices that are part of the sacrificial GEMS installation, including the GEMS PC. System administrators could then reinstall all of the software on the sacrificial GEMS installation, in preparation for the next election. This ensures a clean copy of the GEMS software. Unfortunately, there seems to be no reliable way to clean the sacrificial AV-TSX machines, so reformatting the sacrificial GEMS PC may not be worth the effort.

[1] This is a term applied when two networks are kept physically separate to ensure that data cannot flow from one network to the other. In particular, one ensures that no device attached to the first network is connected (directly or indirectly) to the second network.

Security analysis This architecture prevents viruses introduced by outsiders or poll workers from spreading rapidly over the course of two elections, as described in Section 3.1. While every device in the sacrificial GEMS installation can be easily infected by viruses, this does not help the virus to spread further. The sacrificial GEMS installation is only used for tabulating votes, an operation that can be easily cross-checked during the official canvass by reference to the summary tapes printed in the polling site. The architecture makes it difficult for a virus introduced by an outsider to infect the permanent GEMS installation.

This architecture does not address viruses introduced by insiders. It also does not address slowly spreading viruses. The architecture would need to be supplemented with some additional practices, such as:

- No machine that has ever been used in the field or with the sacrificial GEMS installation should ever be connected to the permanent GEMS installation. Several AV-TSX and AV-OS units should be set aside for use as central-office units to be connected to the permanent GEMS installation, and they should never be used for any other purpose.
- Memory cards and other media would have to be tracked very carefully to ensure that they were never inadvertently inserted into the permanent GEMS installation, as doing so even just once could permanently infect the permanent GEMS installation. For instance, one might institute color-coded labelling and locate the permanent GEMS installation in a physically remote location to reduce the likelihood of accidental procedural lapses.
- If accumulation within a polling site is used, the assignment of AV-OS and AV-TSX machines to each polling place should be permanent. If two machines are assigned to the same polling place in one election, election administrators should avoid assigning them to separate polling places in another election, as that can enable the spread of viruses. If polling-site accumulation is used, then the presumption should be that if any machine in the polling site is infected, malicious code can propagate to infect all of the machines in the polling site. A static assignment of machines to polling sites limits the spread of viruses.
- If modems are used, they should be connected to the sacrificial GEMS installation. No component of the permanent GEMS installation should ever be connected to modems or any other shared network.

6.11 The Alternative: A Voting System that is Secure by Design

Given the costs of designing a new voting system, leaving the Diebold software largely unmodified and relying on procedural changes to mitigate the threats that we describe may seem attractive to policymakers. We consider this to be a risky approach, however, because we are not convinced that it is possible to fully resolve the security problems in the Diebold system through procedural means. We are concerned that, because the Diebold system is vulnerable in so many ways, the procedures needed to protect it would be extensive, complex, and hard to follow. We worry that despite the best efforts and intentions of election officials, the procedures would not be followed perfectly every time and the system would sometimes be left open to attack. As a result, we believe that rather than attempting to retrofit security onto a flawed system, it is safer to reengineer the Diebold system so that it is secure by design.

Building a secure voting system requires making security a central part of the design process from the start. It also demands the involvement of election administrators, experienced software architects and developers, and experts in software security and physical security. Such a system would need to use design techniques appropriate for security-critical systems, such as threat modeling, attack surface reduction, defense in depth, and privilege separation. It would need to apply sound, generally accepted engineering practices for secure software, including input validation, defensive programming, and security testing and assessment. Designing a secure voting system is an expensive proposition that requires a long-term commitment, but the ultimate benefit of doing so is increased confidence in the electoral process.

CHAPTER 7

Conclusion

Our study of the Diebold source code found that the system does not meet the requirements for a security-critical system. It is built upon an inherently fragile design and suffers from implementation flaws that can expose the entire voting system to attacks. These vulnerabilities, if exploited, could jeopardize voter privacy and the integrity of elections. An attack could plausibly be accomplished by a single skilled individual with temporary access to a single voting machine. The damage could be extensive — malicious code could spread to every voting machine in polling places and to county election servers. Even with a paper trail, malicious code might be able to subtly influence close elections, and it could disrupt elections by causing widespread equipment failure on election day.

We conclude that these problems arose because of a failure to design and build the system with security as a central focus, which led to the inconsistent application of accepted security engineering practices. For this reason, the safest way to repair the Diebold system is to reengineer it so that it is secure by design.

We discussed a number of limited solutions and procedural changes that may improve the security of the system, but we warn that implementing any particular set of technical or procedural safeguards may still be insufficient. Similarly, fixing individual flaws in the system — even all of the issues identified in this report — may not yield a secure voting system because of the possibility that unidentified problems will be exploited. We are also concerned that future updates to the system may introduce new, unknown vulnerabilities or fail to adequately correct known ones. We urge the state to conduct further studies to determine whether any new or updated voting systems are secure.

APPENDIX A

Threat Model

The first step in security analysis of a system is to define the *threat model*. The threat model for a system is intended to describe the goals an attacker might have (e. g., to manipulate the vote count) and the types of attackers that might attempt to attack the system (e. g., voters, poll workers, etc.) as well as the capabilities available to each type of attacker. It is equally important to describe the threats that are out of scope for the analysis. This study was chartered only to consider the security of voting systems proper, not that of California's entire election system, and therefore many possible attacks are outside the scope of this report.

A.1 Reference Model

In order to simplify our analysis, we assume a common reference model, which distills the essential features of all the voting systems involved in this study. The reference model consists of the following components.

In the polling place:

- Management stations (MS)
- Direct recording electronic (DRE) voting machines, attached to Voter-Verifiable Paper Audit Trail (VVPAT) printers
- Paper ballot optical scanner (optical scan) machines

At Election Central:

- An election management system (EMS)
- High-speed paper ballot optical scanners (e. g., for absentee votes)

The election can be thought of as proceeding in three stages (for simplicity, ignore early voting centers):

Pre-voting: Before election day, election officials use the EMS to set up the election. They generate the ballot definition(s) and record them onto media for distribution. During this stage, voting machines are also prepared and distributed to polling places.

Voting: On election day, voters arrive at the polling place, are verified as being permitted to vote, and cast their ballots.

Post-voting: After the polls close, the votes are tallied, the official canvass (including the one percent manual recount) is performed, and the results are certified.

The relationship between these components is shown in Figure A.1.

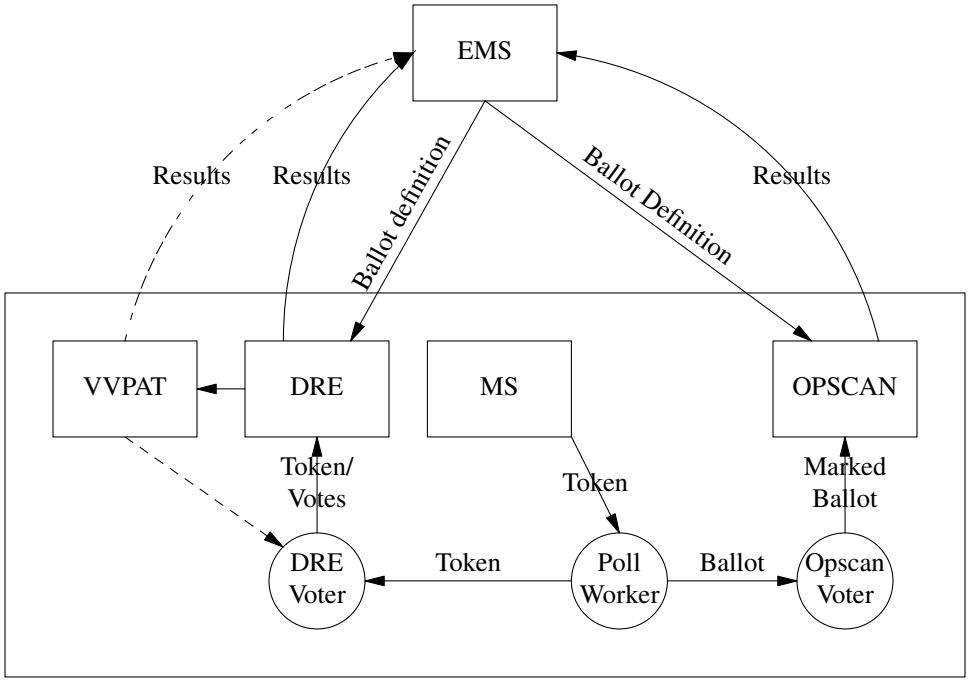


Figure A.1: Reference architecture

A.1.1 Pre-Voting

In the pre-voting phase, election officials need to:

- Create the election definition
- Print the paper ballots used for optical scan systems
- Reset the local voting equipment and load the election definitions
- Distribute the local voting equipment to the polling places

Voting systems vary to a degree, but generally an EMS is used to create ballot definition files. These are then loaded onto some memory cards/cartridges and/or directly onto the voting machines. The vote counters in the machines are reset and the internal clocks are set to the correct time before the machines are shipped out to the local polling places or provided directly to poll workers. To protect ballot definition files from tampering, memory cards/cartridges are generally distributed with physical security measures. These consist of either sealing them into the local equipment at the central office or by distributing them in a sealed package. Seals may take the form of tamper-evident tape or metal or plastic loops which, once installed, can only be removed by cutting them.

A.1.2 Voting

The exact details of the voting phase differ with each technology and manufacturer, but there are common aspects of each technology (DRE or optical scan).

Optical scan machines Optical Scan voting can be thought of as machine-counted paper ballots. When a voter enters the polling place and receives permission to vote, he is given a blank paper ballot. The voter marks the ballot with a pen or pencil and the ballot is then mechanically counted with an optical scanner. This can be done either locally at the precinct or centrally at the county

election headquarters. When done locally, the precinct has a scanner that counts the ballots as they are inserted. Generally, the voter inserts the ballot directly into the scanner. Precinct-based optical scanners can detect "overvoting" and reject such ballots, giving the voter an opportunity to correct the error. When the election ends, the scanner's electronic records are sent back to the county for aggregation with records from other precincts. The paper ballots are also sent back, for auditing and recounts.

With central counting, untabulated ballots are sent back to the county's election headquarters in their original ballot box where they are tabulated with a high-speed scanner under the supervision of election officials. Central- and precinct-based tabulation may be used together in a variety of ways. Central tabulation is better suited for absentee ballots but can also be used for audits and recounts of precinct-cast ballots, whether or not they were originally tabulated in the precinct.

DRE machines DRE voting differs fundamentally from optical scan voting. Instead of entering their vote on paper, voters use a computer-based *graphical user interface* (GUI). Once each vote has been "cast," an electronic record is stored locally, in the DRE machine and possibly in the memory card/cartridge or in another attached system. At the end of the day, these electronic records may either be exported from the DRE machines to memory cards or transmitted via modems. Alternatively, the DREs themselves may be transported to the county's election headquarters. In any case, the EMS will collect the electronic records from each precinct and tabulate them electronically.

In DRE voting (as with optical scan voting), the voter enters the polling place and establishes her eligibility to vote. However, because there is no paper given to the voter, authorized voters must be prevented from casting more than one vote. In every DRE system we analyzed in this study, the poll worker uses an administrative device to issue the voter some type of token. The voter takes this token to any voting machine and is allowed to vote only once. The Diebold and Sequoia systems use a smart card as a token, while the Hart system uses a four-digit "Access Code."

Once the voting machine is activated, a voter can page through each contest and select her choices. DRE machines do not allow overvotes (too many votes cast in a contest) but do allow undervotes (too few choices cast in a contest). The voter is then presented with an opportunity to review the ballot and then commits to it ("casts" it), at which point it is recorded to local storage.

California requires DREs to use a *voter-verifiable paper audit trail* (VVPAT). On all the machines we studied, this takes the form of a sealed printer attached to the DRE that contains a continuous spool of paper. Before the voter confirms the ballot, the machine prints out a paper ballot record and displays it to the voter. Once the voter accepts or rejects the ballot, an appropriate indication is printed on the VVPAT record. When the voter casts his ballot, the record is marked as accepted and scrolled out of sight. Because the paper is displayed behind a glass panel, a voter cannot easily "stuff" additional ballots into the machine or take the record of their vote home as a "receipt."

When the election is over, the local results are transmitted to the county election headquarters, typically by shipping a removable memory device from the voting machine. The VVPAT paper rolls, perhaps still sealed in their printers, are also sent to the county for use in audits.

Typical deployments There are two common models for deploying this equipment in the polling place, DRE-only and a hybrid model. In the DRE-only model, every polling place contains one or more DRE machine and most or all voters vote on the DREs. The polling place contains no optical scan machines. In the hybrid model, every polling place contains one optical scan machine and one or more DREs. Voters either have the option of voting on paper ballots or using the DRE, or the DRE may be reserved for voters with disabilities while all others vote on paper ballots. In California, each county determines which model is most appropriate for its needs.

A.1.3 Post-Voting

After the election is over, the election officers need to do (at least) three things:

1. Tabulate the uncounted optical scan and absentee ballots

2. Produce combined tallies for each contest based on the records received from the individual precincts and the tallies of centrally counted ballots

3. Perform the official canvass. This may involve reconciling the number of voters who have signed in against the number of ballots cast, performing the statutory 1% manual recount, and other tasks

It's important to note that the second task is typically performed based on electronic records. In the most common case, precincts send back memory cards containing election results which are added directly to the tally without any reference to the paper trail (except, of course, for centrally tabulated optical scan and absentee ballots).

The 1% manual recount compares the paper records of a given set of votes to the reported vote totals. In the case of optical scan ballots, this means manually assessing each optical scan ballot. In the case of DREs, it means manually assessing the votes on the VVPAT records. Note that while in principle the optical scan tally may differ slightly from the paper ballots due to variation in the sensitivity of the mark/sense scanner, the VVPAT records should exactly match the DRE records.

A.2 Attacker Goals

At a high level, an attacker might wish to pursue any of the following goals or some combination thereof:

- Produce incorrect vote counts
- Block some or all voters from voting
- Violate the secrecy of the ballot

An attacker might wish to target specific types of voters or the populace at large. For instance, an attacker might target voters who are registered to a particular party, or voters who live within a certain geographic area for attack. The ability to determine how an individual voted also can be used to enable vote buying or voter coercion, either individually or en masse.

A.2.1 Producing Incorrect Vote Counts

The most obvious attack on a voting system is to produce incorrect vote counts. An attacker who can cause the votes to be recorded or counted in a way that is different from how people actually voted can alter the outcome of the election. There are a number of different ways to influence vote counts, including:

- Confuse voters into voting differently than their intent
- Alter the votes, within the computer, before they are recorded
- Alter votes in the vote storage medium, after they were originally recorded
- Corrupt the vote tabulation process

Whether or not specific attacks are feasible depends on attackers' capabilities.

A.2.2 Blocking Some or All Voters from Voting

Two classical techniques to influence election outcomes, regardless of the election technologies in use, are voter education/encouragement (i. e., "get out the vote") and voter suppression. An attacker might:

- Arrange for some subset of machines to malfunction

- Arrange for machines to selectively malfunction when voters attempt to vote in a certain way
- Arrange for all the machines in an election to malfunction

The first two attacks are selectively targeted and could be used to influence the outcome of an election. The third, more global attack is primarily useful for invalidating an election, but could also potentially be used for extortion.

A.2.3 Violating Ballot Secrecy

An attacker unable to influence voting directly might still be able to determine how individuals or groups voted. He might engage in either:

- Vote buying and voter coercion
- Information gathering

In a vote buying attack, the attacker pays (or threatens) individual voters to vote in a specific way. In order for this attack to be successful, the attacker needs to be able to verify how the voter voted. Note that the attacker does not need to be absolutely certain how the voter voted—he must only make the voter believe that the voter's vote has a good chance of being verified.

Another reason an attacker might violate voter secrecy might be to gather information about a large group of people. For instance, an attacker might wish to determine which voters were sympathetic to a particular political party and target them for investigation or surveillance. Alternately, an attacker might wish to publish a prominent voter's votes in order to influence public opinion. In either case, ballot secrecy is required to block this attack.

A.3 Attacker Types

Different kinds of attackers can attack the voting system in different ways. We consider the following broad classes of attacker, listed roughly in order of increasing capability.

Outsiders: have no special access to any of the voting equipment. To the extent that voting or tabulation equipment is connected to the Internet, modems, wireless technologies, and so forth, an attacker can mount network or malware-based attacks. Outsiders may also be able to break into locations where voting equipment is stored unattended and tamper with the equipment.

Voters: have limited and partially supervised access to voting systems during the process of casting their votes.

Poll workers: have extensive access to polling place equipment, including management terminals, before, during, and after voting.

Election officials: have extensive access both to the back-end election management systems as well as to the voting equipment that will be sent to each precinct.

Vendor employees: have access to the hardware and source code of the system during development and may also be called upon during the election process to assist poll workers and election officials.

Note that these categories are not intended to be mutually exclusive — an attacker might have the capabilities of more than one category.

A common focus of security analysis is *privilege escalation*. In many cases, one participant in the system is forbidden from performing actions which can be performed by another participant in the system. A key feature of a secure design is enforcing such restrictions. For instance, a voter should only be allowed to vote once, but poll workers are able to authorize new voters to access the system.

A voter able to use his access to the voting terminal authorize new voters would be an example of someone obtaining privilege escalation. Similarly, poll workers are entrusted with maintaining the integrity of their polling place. If a poll worker were able to use his access to one polling place to influence or disrupt the voting equipment located in other polling places, that would be another example of privilege escalation.

A.3.1 Outsiders

An outsider has no authorized access to any piece of voting equipment. He may be physically separated from the system or present but unable to physically touch the equipment. An outsider has limited capabilities in the context of this review. They might, for instance, enter the polling place with guns and forcefully manipulate the voting systems (at least until the police arrive). This sort of attack is explicitly out of our scope, although the "booth capture" problem is very much a real concern outside the U.S. [28].

Outsiders may have the power to mount network- or malware-based attacks. Both election management systems and the development systems used by the voting vendors typically run on general purpose operating systems (Windows, in the case of all the systems in this review). If those machines are connected to the Internet, either directly or indirectly, an attacker might manage to infect the systems and alter the software running on the machines. If so, any individual with an Internet connection would have the opportunity to attack the voting system.

Outsiders may also have the power to physically tamper with voting equipment. In many counties, voting equipment is stored unattended at the polling place the night before the election. While polling places may be locked overnight, most polling places are low-security locations. For example, they may be located at a school, a church, a public building, or a citizen's garage. Consequently, an attacker able to break into the polling place can likely obtain unsupervised physical access to the voter equipment for several hours.

An outsider may also be able to impersonate other roles in the system, such as a vendor representative or an election official. For example, an outsider might mail a CD containing a malicious software upgrade to the election official in packaging closely resembling the official packaging from the vendor.

A.3.2 Voters

An attacker able to register to vote would be able to launch voter attacks. Unlike an outsider, a voter has physical access to voting machines for at least a short period of time. Because this access is partially supervised, we would not expect a voter to be able to completely disassemble the voting machine. However, in order to preserve the secrecy of the ballot, it is also partially unsupervised. The details of the level of supervision vary to some extent from machine to machine and from county to county. In particular, the difference between optical scan machines and DREs is relevant here.

Optical scan machines In optical scan voting, the voter marks the ballot, which is a special type of paper, herself. The voter then inserts the ballot into the optical scan machine, typically under the supervision of a poll worker. Therefore, the voter is likely unable to tamper with the scanner without being detected. This does not entirely preclude voter access to open I/O ports on the scanner but does make an attack more difficult. The most likely avenue of voter attack is by the ballot itself. For instance, a voter might mark malicious patterns on the ballot intended to subvert the scanner or might attempt to vote multiple times. Such specific patterns could also be used to trigger a dormant "Trojan horse" to cause a compromised machine to begin cheating.

DRE machines In DRE voting, by contrast, the voter has mostly unsupervised access to the voting terminal for a short period of time. The front of the terminal is hidden by a privacy screen in order to protect voters' secrecy, allowing voters to mount a variety of attacks. Any aspect of the machine that is accessible to a voter inside the privacy screen, including buttons, card slots and open I/O

ports, must be assumed to be a potential point of attack. The voter also has an opportunity to input data into the system via the normal user interface. It may be possible to use this interface to compromise the machine.

In all the systems studied here, the voter is provided with some kind of token used to authorize access to the DRE terminal to accept the voter's vote. In the Sequoia and Diebold systems this is a smart card and in the Hart system it is a four-digit access code. As the voter has access to these tokens, the tokens are also a potential target of attack as the voter might attempt to substitute a counterfeit token or subvert the smart card.

A.3.3 Poll Workers

Local poll workers have a significant capability that voters do not: legitimate access to the management functions of the equipment. For instance, the poll worker has the ability to authorize voters. Note that although in principle this may give the poll worker opportunities for malfeasance, this risk may be mitigated by procedural controls. For instance, a poll worker who controls the management station can in principle authorize a voter to vote an arbitrary number of times simply by issuing multiple tokens. However, polling places would normally have procedures in place to block or at least detect such attacks. Poll workers must perform their duties in public view, so any such malfeasance might be noticed by other poll workers or other voters. Moreover, if at the end of the day there were more votes cast than registered voters who signed in, officials would investigate the cause of the discrepancy. Purely technical means may, alone, be insufficient to prevent such attacks, but procedural mechanisms may be sufficient to address the risks.

Depending upon county practices, poll workers may also have long-term unsupervised access to voting equipment. In some counties, voting equipment is stored in the houses or cars of individual poll workers prior to the election. For instance, some counties provide the chief poll worker at each polling place machines to store and deliver to the polling place. Even counties that deliver machines by commercial transport may provide the chief poll worker with other equipment (smart cards, smart card activation devices, management consoles, etc.) before the election. Even if equipment is stored in a secured polling place, controls may not be in place to prevent individual poll workers from accessing the area on their own. This provides a number of opportunities for tampering with equipment. Many pieces of equipment include seals to detect such tampering, but each system must be individually analyzed to determine whether these seals are effective and whether they protect all the relevant access points.

It must be noted that poll workers are primarily volunteers and are subject to extremely minimal security screening, if any is performed at all. In many counties the need for poll workers is so great that any registered voter who calls and offers to serve sufficiently far in advance is almost sure to be hired, and poll workers are often allowed to request to serve at a particular precinct. In practice we must assume that any attacker who wants will be able to be a poll worker.

A.3.4 Election Officials

County election officials and staff have three significant capabilities that poll workers do not:

- Access to functionality of local voting equipment which may be restricted from poll workers
- Access to large amounts of local voting equipment between elections
- Access to the back-end election management system used for equipment management, ballot creation and tabulation

For reasons of administrative efficiency, this access might be unsupervised or only loosely supervised, depending upon county practices.

The first two capabilities imply greater ability to mount the kinds of attacks that poll workers can mount. An election official with access to the warehouse where voting machines are stored might be able to compromise all the equipment in a county rather than merely all the machines in a precinct. In addition, the procedures for some equipment require that they be sealed — for

instance, the memory cards or results cartridges may be sealed inside the machine — before they are sent to the precincts. Because this sealing happens under the supervision of election officials, those officials might be able to bypass or subvert that process.

The third capability is wholly unavailable to the local poll worker. The back-end election management systems typically run on general purpose computers which are used by the election officials. If those systems are subverted they could be used to compromise polling place voting equipment, create fake or incorrect ballots, and to miscount votes. This subversion could happen in at least three ways. First, many attacks can be mounted using only the attacker's authorized access and within the context of the technical controls which the systems are expected to enforce. For example, the software may offer an opportunity for election official to make "corrections" to vote tallies. Such "corrections" might be incorrect. Second, an election official might find a way to defeat the technical access controls in the election management software and tamper with its vote records. Third, the official could directly subvert the computers on which the software runs. Experience with computer security shows that an attacker who has physical access to a general purpose computer can almost always take control of it. This may be achieved in a number of ways ranging from software attacks to directly compromising the system hardware. The clear implication of this fact is that if election officials have unsupervised access to the election management systems, the integrity of those systems is provided mainly by procedural controls and the honesty of officials, not by any technical measures.

A.3.5 Vendor Employees

Finally, we consider attackers in the employ of the vendors. Such attackers fall into two categories: those involved in the production of the hardware and software, prior to the election, and those present at the polling place or Election Central warehouse during an election. An individual attacker might of course fall into both categories.

An attacker involved in the development or production of the software and hardware for the election system has ample opportunities to subvert the system. He might, for instance, insert malicious code into the election software, insert exploitable vulnerabilities or back doors into the system, or design the hardware in such a way that it is easily tampered with. Such attacks are extremely hard to detect, especially when they can be passed off as simple mistakes or bugs as those are extremely common in large software projects. It may be very difficult to distinguish good-faith mistakes from deliberate subversion. Note that for such an attack to succeed it is not necessary for the attacker to arrange for uncertified software or hardware to be accepted by election officials or poll workers. Rather, the vulnerabilities would be in the certified versions. Neither the current certification process nor this review is intended or able to detect all such vulnerabilities.

A vendor employee may also be present in the county to assist election officials or poll workers. For instance, the employee might be present at Election Central during or after the election to help election officials, either by answering questions or by helping to fix or work around malfunctioning equipment. A vendor employee might also be present at Election Central to help install or maintain the voting system or to train county staff or poll workers. A vendor employee might even be present at the polling place or available by phone to assist poll workers or answer questions. Such an attacker would have access to equipment comparable to that of poll workers or election officials, but would also have substantial freedom of movement. Because they are being asked to correct malfunctions and install and configure software, activities which are actually intended to subvert the equipment are much less likely to be noticed. To the extent to which the systems have hidden administrative interfaces they would presumably have access to those as well. Finally, vendor employees pose a heightened risk because they may have access to multiple counties which use the vendor's equipment, and the ability of one individual to able to tamper with voting equipment in multiple counties increases the scope of any potential subversion.

A.4 Types of Attacks

We can categorize attacks along several dimensions:

- *Detectable vs. undetectable.* Some attacks are undetectable: they cannot be detected, no matter what practices are followed. Others are detectable in principle, but are unlikely to be detected by the routine practices currently in place; they might be detected by an in-depth forensic audit or a 100% recount, for instance, but not by ordinary processes. Still other attacks are both detectable and likely to be detected by the practices and processes that are routinely followed.

  The potential harm caused by the former two classes of attacks surpasses what one might expect by estimating their likelihood of occurrence. The mere existence of vulnerabilities that make likely-to-be-undetected attacks possible poses a threat to election confidence. If an election system is subject to such attacks, then we can never be certain that the election results were not corrupted by undetected tampering. This opens every election up to question and undercuts the finality and perceived fairness of elections. Therefore, we consider undetectable or likely-to-be-undetected attacks to be especially severe and an especially high priority.

- *Recoverable vs. unrecoverable.* In some cases, if an attack is detected, there is an easy way to recover. In contrast, other attacks can be detected, but there may be no good recovery strategy short of holding a new election. In intermediate cases, recovery may be possible but expensive (e. g., recovery strategies that involve a 100% manual recount impose a heavy administrative and financial burden).

  Attacks that are detectable but not recoverable are serious. Holding a new election is an extreme remedy, often requiring contentious litigation. Also, unofficial election results, once announced, tend to take on certain inertia and there may be a presumption against abandoning them. When errors are detected, attempts to overturn election results can potentially lead to heated partisan disputes. Even if errors are detected and corrected, a failure can potentially diminish public confidence. At the same time, detectable-but-not-recoverable attacks are arguably not as serious as undetectable attacks: we can presume that most elections will not be subject to attack, and the ability to verify that any particular election was not attacked is valuable.

- *Wholesale vs. retail.* One can distinguish attacks that attempt to tamper with many ballots or affect many voters' votes ("wholesale" attacks) from attacks that attempt to tamper with only a few votes ("retail" attacks). For instance, attacks that affect an entire county or a large fraction of the precincts within a county are typically classified as wholesale attacks, whereas attacks that affect only one voter or one precinct are typically classified as retail attacks. This is a useful distinction because, in most contests, retail fraud does not have an impact large enough to change the outcome of an election. Because wholesale fraud has a more significant impact, we focused especially on analyzing whether the systems are vulnerable to wholesale fraud.

- *Casual vs. sophisticated.* Certain attacks require little technical knowledge, sophistication, advance planning, resources, or access. For instance, stealing an absentee ballot out of someone's mailbox is a classic low-tech attack: anyone can execute such an attack, and no special qualifications or skills or organization is needed. In contrast, other attacks may require deep technical knowledge, specialized skills or expertise, considerable advance planning, a great deal of time, money, or other resources, and/or insider access. This study examines both sophisticated technical attacks as well as casual low-tech attacks.

We can also categorize defenses:

- *Prevention vs. detection.* Often, there is a tradeoff between different strategies for dealing with attacks. One strategy is to design mechanisms to prevent the attack entirely, closing

the vulnerability and rendering attack impossible. When prevention is not possible or too costly, an attractive alternative strategy is to design mechanisms to detect attacks and recover from them.

Most election systems combine both strategies, using prevention as the first line of defense along with detection as a fallback in case the preventive barrier is breached. For instance, we attempt to prevent or deter ballot box stuffing by placing the ballot box in the open where it can be observed. At the same time, we track the number of signed-in voters, account for all blank ballots, and count the number of ballots in the box at the end of the day to ensure that any ballot box stuffing that somehow escapes notice will still be detected. This combination can provide a robust defense against attack.

A.5 Mechanisms for Tamper Sealing

Virtually every election system makes extensive use of tamper seals as a part of its security design. This section presents a brief summary of how these mechanisms work and the level of sophistication an attacker must have to violate them.

Tamper resistance refers to the ability of a system to deter an attacker from gaining access to the system. This could take the form of software controls (e. g., careful limits on the protocols spoken across networks) to procedural controls (e. g., the use of strong passwords) to hardware mechanisms (e. g., strong locks). Tamper resistance generally refers to the amount of time, effort, and/or sophistication required to overcome a security mechanism.

Tamper evidence is the flip-side of the coin to tamper resistance, representing the extent to which an attacker's attempt to overcome a tamper-resistance mechanism leaves behind evidence of that tampering. For example, in a typical home, a burglar could easily break in by putting a brick through a window. A plate-glass window offers little tamper resistance, but strong tamper evidence (i. e., the effort that would be required by a burglar to reinstall a broken window and clean up the broken glass is quite significant). For contrast, a typical door lock is far more tamper resistant than the glass window. However, if a skilled burglar can pick the lock, there will be little or no evidence that it had been picked.

In the context of voting systems, there are a number of tamper sealing mechanisms commonly used:

Key locks: To prevent access to memory cards or sensitive machine ports, many voting machines place a plastic or metal door in front of these ports, using a key lock. Assuming the keys are suitably controlled (and unauthorized duplication is prevented), attackers would be prevented from accessing the protected ports. Of course, if bypassable lock mechanisms are used, or if access to the locked compartment can be gained without opening the lock, then the locks will offer neither tamper resistance nor tamper evidence as has been observed with both Diebold [14] and Nedap/Groenendaal [15] voting systems.

Wire loops: Many voting machines have adopted a mechanism commonly used with traditional ballot boxes — the use of holes through which a metal or plastic wires loops may be fitted. These seals have much in common with standard "tie wraps;" once fitted, the wire loop cannot be loosened; it can only be physically cut off. Like key locks, the loops, when sealed, lock a physical door in place. In common election practice, these seals are stamped or printed with individual serial numbers. Those numbers are then logged when the seals are installed and again when they are cut to detect the substitution of an alternate seal. An attacker with simple tools may be able to clone the serial numbers from old wire loops to new ones without detection [31].

Tamper-evident tape: Adhesive tape can be printed with numbered labels in the same fashion as loop seals. Typically, two different adhesives are used, such that if/when the tape is removed, part of the label will remain stuck to the lower surface while part of the label will be removed with the tape. Technology of this sort is commonly used for automobile registration and

inspection stickers. Anecdotal evidence suggests that it may be possible to peel back the tape and replace it without this being easily observable [29].

A recent study of tamper seals considered 244 different seal designs and found that "the majority could be defeated — removed and replaced without evidence — by one person working alone within about two minutes and all of these devices could be thwarted within about 30 minutes" [20]. Needless to say, such seals cannot be counted on, alone, to provide significant security protections for electronic voting systems.

Of course, these mechanisms can be augmented through other procedural means, including requiring multiple people to be present when machines are handled or maintaining video cameras and other locks on the storage areas of the elections warehouse. Johnston also recommends that officials have genuine seals in their hands to compare against the seals being inspected [20].

The use of tamper-evident or tamper-resistant technologies, as such, must be evaluated in the broader context of procedures and policies used to manage an election. Weaknesses in these procedures cannot be overcome by the application of tamper-resistant / tamper-evident seals. Also, the attacker's motivation must also be considered. Perhaps the attacker does not care if an attack is evident, so long as it cannot be recovered from.

Bibliography

[1] Interview with Talbot Iredale, Software Development Manager, Diebold Election Systems.

[2] Voting machine company apologizes to Kern County. *KERO 23 Bakersfield*. June 13, 2006. Available at http://www.turnto23.com/news/9366552/detail.html.

[3] ABBOT, R. P., DAVIS, M., EDMONDS, J., FLORER, L., PROEBSTEL, E., PORTER, B., AND STAUFFER, J. Security evaluation of the Diebold voting system, July 2007.

[4] BRENNAN CENTER TASK FORCE ON VOTING SYSTEM SECURITY. *The Machinery of Democracy: Protecting Elections in an Electronic World*, June 2006. Available at http://www.brennancenter.org/dynamic/subpages/download_file_36343.pdf.

[5] BURREN, D. man crypt(3) (FreeBSD 6.2). Available at http://www.freebsd.org/cgi/man.cgi?query=crypt&sektion=3, Jan. 1997. Multiple editors.

[6] CHRISTEY, S., AND MARTIN, R. A. Vulnerability type distributions in CVE, version 1.1. Available at http://cwe.mitre.org/documents/vuln-trends/index.html, May 2007.

[7] CHUNG, L. Microsoft Access or Microsoft SQL Server: What's right in your organization. Available at http://download.microsoft.com/download/5/d/0/5d026b60-e4be-42fc-a250-2d75c49172bc/Access_Whats_Right.doc, Dec. 2004.

[8] DIEBOLD ELECTION SYSTEMS. Frequently asked questions. Available at http://www.diebold.com/dieboldes/faq.asp.

[9] DIEBOLD ELECTION SYSTEMS. AccuVote-OS hardware guide revision 6.0, Feb. 2005.

[10] DIEBOLD ELECTION SYSTEMS. GEMS 1.18 election administrator's guide 8.0, June 2005.

[11] DILL, D., AND WALLACH, D. Stones unturned: Gaps in the investigation of Sarasotas disputed congressional election. Available at http://www.cs.rice.edu/~dwallach/pub/sarasota07.html, Apr. 2007.

[12] DRIEHAUS, B. Audit finds many faults in Cleveland's '06 voting. *The New York Times*. April 20, 2007.

[13] EVERETT, S. P. *The Usability of Electronic Voting Machines and How Votes Can Be Changed Without Detection*. PhD thesis, Rice University, 2007.

[14] FELDMAN, A., HALDERMAN, J. A., AND FELTEN, E. W. Security analysis of the Diebold AccuVote-TS voting machine. In *Proc. 2007 USENIX/ACCURATE Electronic Voting Technology Workshop (EVT '07)*.

[15] GONGGRIJP, R., AND HENGEVELD, W.-J. Studying the Nedap/Groenendaal ES3B voting computer: A computer security perspective. In *Proc. 2007 USENIX/ACCURATE Electronic Voting Technology Workshop (EVT '07)*.

[16] HOGLUND, G., AND BUTLER, J. *Rootkits: Subverting the Windows Kernel*. Addison-Wesley Professional, 2005.

[17] HURSTI, H. Critical security issues with Diebold optical scan design. Available at http://www.blackboxvoting.org/BBVreport.pdf, July 2005.

[18] HURSTI, H. Critical security issues with Diebold TSx (unredacted). Available at http://www.wheresthepaper.org/reports/BBVreportIIunredacted.pdf, May 2006.

[19] HURSTI, H. Diebold TSx evaluation: Supplemental report, additional observations (unredacted). Available at http://www.wheresthepaper.org/BBVreportIIsupplementUnredacted.pdf, May 2006.

[20] JOHNSTON, R. G. Tamper-indicating seals. *American Scientist 94* (November-December 2006), 515–523. Reprint available at http://ephemer.al.cl.cam.ac.uk/~rja14/johnson/newpapers/American%20Scientist%20(2006).pdf.

[21] JOHNSTON, R. G. Tamper-indicating seals. *American Scientist 94* (Nov. 2006), 515–523.

[22] JONES, D. W. Voting system transparency and security: The need for standard models, September 2004. Testimony before the EAC Technical Guidelines Development Committee, National Institute of Standards and Technology, Gaithersburg, MD.

[23] KIAYIAS, A., MICHEL, L., RUSSELL, A., AND SHVARTSMAN, A. Security assessment of the Diebold optical scan voting terminal. Available at http://voter.engr.uconn.edu/voter/OS-Report_files/uconn-report-os.pdf, Oct. 2006.

[24] KIAYIAS, A., MICHEL, L., RUSSELL, A., AND SHVARTSMAN, A. Integrity vulnerabilities in the Diebold TSX voting terminal. Available at http://voter.engr.uconn.edu/voter/OS-TSX-Report_files/TSX_Voting_Terminal_Report.pdf, July 2007.

[25] KLEIN, D. V. Foiling the cracker: A survey of, and improvements to, password security. In *Proc. 1990 USENIX Workshop on Security*.

[26] KOHNO, T., STUBBLEFIELD, A., RUBIN, A., AND WALLACH, D. Analysis of an electronic voting system. In *Proc. 2004 IEEE Symposium on Security and Privacy*, pp. 27–42.

[27] PELZL, J. Cryptanalysis with a cost-optimized FPGA cluster (presentation slides). UCLA IPAM Workshop IV: Special purpose hardware for cryptography: Attacks and Applications. Available at https://www.ipam.ucla.edu/publications/scws4/scws4_6560.pdf, Dec. 2006.

[28] ROHDE, D. On new voting machine, the same old fraud. *The New York Times*. April 27, 2004. Available at http://www.nytimes.com/2004/04/27/international/asia/27indi.html.

[29] RUBIN, A. D. My day at the polls — Maryland primary '06. Available at http://avi-rubin.blogspot.com/2006/09/my-day-at-polls-maryland-primary-06.html, Sept. 2006.

[30] RYAN, T. P., AND HOKE, C. GEMS tabulation database design issues in relation to voting systems certification standards. In *Proc. 2007 USENIX/ACCURATE Electronic Voting Technology Workshop (EVT'07)*.

[31] SHAMOS, M. Oral testimony, Technical Guidelines Development Committee (TGDC), public data gathering hearings. Available at http://vote.nist.gov/PublicHearings/9-20-94%20Panel%202%20SHAMOS.doc, Sept. 2004.

[32] SUN MICROSYSTEMS. Java card technologies. Available at http://java.sun.com/products/javacard/.

[33] WAGNER, D., JEFFERSON, D., AND BISHOP, M. Security analysis of the Diebold AccuBasic interpreter. Available at http://www.ss.ca.gov/elections/voting_systems/security_analysis_of_the_diebold_accubasic_interpreter.pdf, Feb. 2006.

[34] YOUNG, E. des(3). Available at http://www.openssl.org/docs/crypto/des.html.

Overview of Red Team Reports

Matt Bishop, Principle Investigator, University of California, Davis

1.0. Executive Summary

The California Secretary of State entered into a contract with the University of California to test the security of three electronic voting systems as part of her top to bottom review. Each "red team" was to try to compromise the accuracy, security, and integrity of the voting systems without making assumptions about compensating controls or procedural mitigation measures that vendors, the Secretary of State, or individual counties may have adopted. The red teams demonstrated that, under these conditions, the technology and security of all three systems could be compromised.

2.0 Goals

In May 2007, the California Secretary of State began a study of all electronic voting systems currently certified in California. This "top to bottom review" (TTBR) was to determine whether the systems currently certified should be left alone, or specific procedures required to provide additional protections for their use, or the machines simply decertified and banned from use. As part of this study, the Secretary contracted with the University of California to conduct a "red team" review of the systems. The specific goal of the Red Team study was "to identify and document vulnerabilities, if any, to tampering or error that could cause incorrect recording, tabulation, tallying or reporting of votes or that could alter critical election data such as election definition or system audit data." ([1], p. 5).

A red team study, also called a penetration study, examines a system from the point of view of an attacker, and analyzes the system to determine how secure it is against an attack. Such a study requires establishing several parameters:

- The specific goals of the system: what is it to do?
- The threat model: with whom or what are the testers concerned?
- The information to be made available to the testers: how much do they know at the start?
- The environment in which the system is used: what policies and procedures are to be applied?
- The specific "rules of engagement": what are the team members allowed to do?

For this TTBR, the specific goals of each system are to record, tabulate, tally, and report votes correctly and to prevent critical election data and system audit data from being altered without authorization. The threats were taken to be both insiders (those with complete knowledge of the system and various degrees of access to the system) and outsiders (those with limited access to the systems). As a result, *all* information available to the Secretary of State was made available to the testers. The testers were told to assume that the environments in which the systems were used would vary, and that the

testers could do whatever they thought necessary to test the machines. The testers therefore assumed the attackers would include anyone coming in contact with the voting systems at some point in the process – voters, poll workers, election officials, vendor employees, and others with varying degrees of access [18].

In developing attack scenarios, the red teams made no assumptions about constraints on the attackers. We recommend that future Red Teams should adopt a similar attitude.

The testers did *not* evaluate the likelihood of any attack being feasible. Instead, they described the conditions necessary for an attacker to succeed. This approach had several benefits:

- The testers could focus on the technology rather than on the policies procedures, and laws intended to compensate for any technological shortcomings.
- In California, specific procedures for controlling access to the election systems and for setting up, using, and storing the election systems is a local matter. As there are 58 different counties, there are at least 58 different sets of procedures. It was impractical for the red team testers to evaluate them.
- If a problem is discovered, the people who know the law and election policies and procedures can modify their policies and procedures appropriately to attempt to address the problem.
- Finally, the effectiveness of the policies and procedures used to control and protect the election systems depends on their implementation. Policies and procedures that look effective on paper may be implemented poorly, rendering them ineffective. It was impractical to evaluate this aspect of the policies and procedures.

Therefore, the results of this study must be evaluated in light of the context in which these election systems are used. This emphasizes a key point often overlooked in the discussion of the benefits and drawbacks of electronic voting systems: those systems are part of a process, the election process; and the key question is whether the election process, taken as a whole, meets the requirements of an election as defined by the body politic.

The participants in this study hope our work contributes in some measure to answering that question.

2.1 Systems Examined

Three systems were reviewed in this study.

Diebold. The Diebold GEMS 1.18.24/AccuVote consisted of the following components:

- GEMS software, version 1.18.24
- AccuVote-TSX with AccuView Printer Module and Ballot Station firmware version 4.6.4
- AccuVote-OS (Model D) with firmware version 1.96.6
- AccuVote-OS Central Count with firmware version 2.0.12

- AccuFeed
- Vote Card Encoder, version 1.3.2
- Key Card Tool software, version 4.6.1
- VC Programmer software, version 4.6.1

Hart Intercivic. The Hart Intercivic System 6.2.1 consisted of the following components:

- Ballot Now software, version 3.3.11
- BOSS software, version 4.3.13
- Rally software, version 2.3.7
- Tally software, version 4.3.10
- SERVO, version 4.2.10
- JBC, version 4.3.1
- eSlate/DAU, version 4.2.13
- eScan, version 1.3.14
- VBO, version 1.8.3
- eCM Manager, version 1.1.7

Sequoia. The Sequoia WinEDS version 3.1.012/Edge/Insight/400-C consisted of the following components:

- WinEDS, version 3.1.012
- AVC Edge Model I, firmware version 5.0.24
- AVC Edge Model II, firmware version 5.0.24
- VeriVote Printer
- Optech 400-C/WinETP firmware version 1.12.4
- Optech Insight, APX K2.10, HPX K1.42
- Optech Insight Plus, APX K2.10, HPX K1.42
- Card Activator, version 5.0.21
- HAAT Model 50, version 1.0.69L
- Memory Pack Reader (MPR), firmware version 2.15

2.2 Team Organization

Two red teams were organized. One team, led by Robert P. Abbott, was based in Sacramento at the Secretary of State's secure facility. The second team, led by Giovanni Vigna and Richard Kemmerer, was based at the University of California, Santa Barbara, and came to Sacramento as needed to use the system. The first team examined the Diebold and Hart systems; the second, the Sequoia system.

3.0 Context

Two contexts are relevant: that of the election process, and that of the system certification.

3.1 Computers as Part of a Process

It is commonly accepted that no computer or computer-based system, called an *information technology system*, can be made completely secure. It is also commonly accepted that the managers of an information technology system have a responsibility to

develop sufficient controls in and around a system to the point that continued operation of the system meets the requirements of the organization. "Organizations should satisfy the quality, fiduciary and security requirements for their information, as for all assets." ([2], p. 5) "A high level of security management may have to be focused only on the most critical enterprise systems." ([2], p. 20)

Electronic voting systems are special purpose computer systems. As such, they require compensating controls just like any other computer system. Protecting computer systems embodies several topics, which, taken together, constitute an Information Technology Security Program. There are many reference guides toward establishing such a program, such as FIPS PUB 200 [7].[1]

An Information Technology Security Plan includes three topics of particular interest to owners of electronic voting systems:

1. *Physical security*. Electronic voting machines must be protected against unauthorized physical and electronic access while: a) in storage; b) at polling locations; c) at the central election center; and d) while in transit between storage, the polls, and the election center.
2. *Security training of staff*. Election officials and poll workers must be acquainted with the concepts of information technology security as well as procedures to invoke when security rules are violated. Training must also address the concept of social engineering, which is a collection of techniques used to manipulate people to perform actions that are forbidden, or divulge information that should remain confidential. An example is pretexting, a technique in which an investigator obtains information about someone's records by pretending to be that person. In 2006, investigators used this technique to try to determine the source of leaks from the HP Board of Directors [17].
3. *Contingency planning*. Plans must be developed to handle the situation in which a polling place or a voting station is rendered inoperative. Every contingency must be thought of and thought through in advance. A work-around process or procedure must be developed and tested for each contingency.

Many, but not all, of the attack scenarios contained in these reports would be mitigated by fully addressing these three topics. The feasibility of developing policies and procedures that can be effectively implemented, what those policies and procedures should be, and how they should be implemented, is a matter that lies within the knowledge and experience of election officials and the California Secretary of State.

Security traditionally relies on layers of mechanisms; this is called *defense in depth*, *layered defense*, or *separation of privilege*. The idea is to force an attacker to breach several security mechanisms to compromise the system, rather than one. Procedures form

[1] Other examples are ISO 9001:2000 [3], the CMMI [4], PRINCE2 [5], and PMBOK [6]. Organizations such as SANS (http://www.sans.org) and (ISC)^2 (http://www.isc2.org) provide training and education on information security practices, also.

some of these layers of defensive mechanisms. Proper system configuration and implementation form additional layers of defensive mechanisms. Security plans should *always* rely on multiple layers. In particular, that procedures could mitigate or block the attack scenarios in this report in no way relieves vendors of their responsibility to locate, repair, and fix the vulnerabilities in their products that these attacks exploit.

Finally, no security should *ever* rely solely on secrecy of defensive mechanisms and countermeasures.[2] While not publishing details of security mechanisms is perfectly acceptable as one security mechanism, it is perhaps the one most easily breached, especially in this age of widespread information dissemination. Worse, it provides a false sense of security. Dumpster diving, corporate espionage, outright bribery, and other techniques can discover secrets that companies and organizations wish to keep hidden; indeed, in many cases, organizations are unaware of their own leaking of information. A perhaps classic example occurred when lawyers for the DVD Copyright Control Association sued to prevent the release of code that would decipher any DVD movie file. They filed a declaration containing the source code of the algorithm. One day later, they asked the court to seal the declaration from public view—but the declaration had been posted to several Internet web sites, including one that had over 21,000 downloads of the declaration! [9] More recently, Fox News reported that information posing "a direct threat to U.S. troops … was posted carelessly to file servers by government agencies and contractors, accessible to anyone online" [8], and thefts of credit card numbers and identities are reported weekly and growing in number. Thus, the statement that attackers could not replicate what red team testers do, because the red team testers have access to information that other attackers would not have, profoundly underestimates the ability and the knowledge of attackers, and profoundly overestimates the infallibility of organizations and human nature.

3.2 Certification

The California Secretary of State must certify any electronic voting system before it can be used in California elections. One of the requirements is that the system be federally certified to meet the 2002 Voting System Standards (VSS) [10]. Independent testing authorities (ITAs) test the electronic voting system to certify compliance with these standards. All three systems in this study were so certified [11,12,13].

The quality of the 2002 standards is inadequate (see Barr *et al*. [14] for an analysis of the 2002 standards and their successor, the 2005 Voluntary Voting System Guidelines [15]). Further, questions have been raised about the effectiveness of the testing. For example, Ciber, Inc., an ITA, has been denied interim accreditation for testing voting systems by the Federal Election Assistance Commission after finding that Ciber "was not following its quality-control procedures and could not document that it was conducting all the required tests" [16].

[2] This is often called "security through obscurity".

4.0 Limits and Problems of the Study

The major problem with this study was time. Although the study did not start until mid-June[3], the end date was set at July 20, and the Secretary of State stated that under no circumstances would it be extended. This left approximately 5 weeks to examine the three systems. For budgetary reasons, the UCSB team (which was examining 1 system) planned to conclude its examination by July 10. In order to do as much as possible, it was decided to examine the Hart and Diebold systems simultaneously, rather than allocate 2.5 weeks to each. The examination of both these systems concluded on July 20.

The short time allocated to this study has several implications. The key one is that ***the results presented in this study should be seen as a "lower bound"; all team members felt that they lacked sufficient time to conduct a thorough examination, and consequently may have missed other serious vulnerabilities***. In particular, Abbott's team reported that it believed it was close to finding several other problems, but stopped in order to prepare and deliver the required reports on time. These unexplored avenues are presented in the reports, so that others may pursue them. Vigna's and Kemmerer's team also reported that they were confident further testing would reveal additional security issues.

The second problem was a lack of information. In particular, various documents did not become available until July 13, too late to be of any value to the red teams, and the red teams did not have several security-related documents.[4] Further, some software that would have materially helped the study was never made available. As a specific example, when installing the system initially, the Hart personnel used a program to upgrade firmware on their system. The red and source code team members present asked for a copy of that program, because it would enable the testers to determine whether anyone could upgrade the firmware[5]. Otherwise, the teams would have to discover the protocol used for upgrading the firmware and write programs to do it themselves. The person doing the installation stated that the program was proprietary and would not be released to the Secretary of State and the teams. The teams asked the Secretary of State to obtain the program from Hart. The request was repeated in a phone call with Hart engineers on July 16, and Hart said they would have to discuss it among themselves. The software was never supplied. The red team and source code review team for Hart worked together and created their own upgrade program that performed suitably for the purposes of testing, but the time they spent doing this could have been spent analyzing other aspects of the system.

---

[3] The Diebold system was set up for use by the testers on June 14, the Sequoia system on June 19, and the Hart system on June 22. The testers were able to photograph the Sequoia system on June 14, but the system was not yet set up for use.

[4] See the reports of the document review teams for details of missing documents and documents that arrived after July 12.

[5] More specifically, if the firmware images were digitally signed, an attacker could not "reflash" (i.e., install) new firmware without having access to the private key. However, if the firmware images were *not* digitally signed, all one would need is access to the system to reflash the firmware and compromise the system—a considerably easier task.

Despite these problems, the red team testing was successful, in that it provided results that are reproducible and speak to the vulnerability of all three systems tested.

5.0 Example Threats

An election system consists of three components: the ballot preparation system, the voting mechanisms, and the tallying systems. The red teams were given sample elections and used them in the elections that they tried to subvert.

The voting mechanisms are either Direct Recording Electronic machines (DREs) with Voter Verified Paper Audit Trails (VVPATs) or optical scan systems. They each store the votes that voters cast in various ways. If they can be compromised, the votes stored on those systems may not reflect the votes actually cast by the voters.

As an example, the ability to execute arbitrary programs on one of these systems can cause votes to be misrecorded even when there is a VVPAT. The specific attack relies on the belief that many voters will not check the VVPAT. An attacker creates a new version of the firmware that will misrecord a vote. The incorrect vote will be printed on the VVPAT. If the voter notices and declines to cast the vote by returning to an earlier screen, the malicious firmware will then record the vote correctly. Thus, there will be no discrepancy between the votes as recorded on the VVPAT and on the electronic media.

Even if there is a discrepancy between the VVPAT and the electronically recorded votes, that discrepancy must be discovered. Typically, this would occur during the 1% audit or a recount. For our purposes, we considered such a discrepancy a valid attack, because the way in which such a discrepancy is to be handled is unclear, especially when the VVPAT is damaged or hard to read.

The election management system consists of software running on a commercial platform. Typically this platform is some form of Microsoft® Windows. The application software consists of a database program and other software. A client program, the database application, or both control access to the election data. This platform may also be used to initialize memory cards or other media to transfer information to the voting machines. The platform may also contain other programs not supplied by the voting system vendor.

For example, all three vendors' election management software runs on platforms with the Windows operating system. The configuration of the Windows system provides a layer of protection against an attacker compromising the software. The strength of this layer depends directly on the security of the underlying operating system. As Windows is known to be vulnerable to many forms of attack, vendors should ensure that the underlying Windows system is locked down sufficiently[6] to counter these threats.

If an attacker can gain privileged access to the underlying operating system, they can control the election management system. This is why election management systems

[6] For example, one step in this procedure would be to disable all unnecessary services.

should be locked down tightly and be kept in a physically secure area: so that attackers have limited to no access to the system. As noted above, physical access is simply one layer of security defense. Minimizing privileges and taking other basic precautions in configuring the underlying operating system provide additional layers.

As an example, if an attacker can insert an untrusted medium (like a U3 USB memory stick) containing a malicious program called a "Trojan horse", and the system is configured so that autorun is turned on, the Trojan horse can be loaded into memory. At that point, it can detect the insertion and removal of media, including media intended to load information onto the voting machine. It can load malicious firmware onto that media. It can modify any local files, and completely control the underlying system.

The results presented in the next section show that the above attacks, and many like them, can be realized.

6.0 Results and Interpretations

This section presents a very high-level overview of the test results, and their technical interpretation. It is up to the Secretary of State, and other election officials, to interpret these findings in light of the relevant laws, regulations, policies, and procedures.

The results are documented in three reports, one for each system. Each report consists of a public portion and a confidential portion. Our goal in the public portion is to provide information about the vulnerabilities of the systems to the public to allow intelligent and reasoned discourse on the effects of those vulnerabilities and on the role of these electronic voting systems in the California election process without providing a step-by-step guide to attacking the systems, and without revealing the vendors' proprietary information. The confidential portion details the attacks that were successful, discusses those attacks that were tried but that failed, and also provides suggestions for attacks that we were unable to try. We hope, and intend, that this material provide guidance to future testers.

We request that the Secretary of State provide the public and confidential reports to the respective vendors. We believe the vendors want to close any vulnerabilities found. We believe that our reports can help them identify those vulnerabilities, how they might be exploited, and how they might be mitigated or eliminated. With their intimate knowledge of their systems, this should be enough to enable them to determine, and take, appropriate corrective action.

6.1 Sequoia

The red team analyzing the Sequoia system identified several issues. They fall into several classes:

1. **Physical Security**. The testers were able to gain access to the internals of the systems by, for example, unscrewing screws to bypass locks. The screws were not protected by seals. Similarly, plastic covers that were protected by seals could be pried open enough to insert tools that could manipulate the protected buttons without damaging

the seals or leaving any evidence that the security of the system had been compromised.

2. **Overwriting Firmware**. The testers discovered numerous ways to overwrite the firmware of the Sequoia Edge system, using (for example) malformed font files and doctored update cartridges. The general approach was to write a program into memory and use that to write the corrupt firmware onto disk. At the next reboot, the boot loader loaded the malicious firmware. At this point, the attackers controlled the machine, and could manipulate the results of the election. No source code access was required or used for this attack, and a feature of the proprietary operating system on the Edge made the attack easier than if a commercial operating system had been used.
3. **Overwriting the Boot Loader**. Just as the testers could overwrite firmware on the disk, they could overwrite the boot loader and replace it with a malicious boot loader. This program could then corrupt anything it loaded, including previously uncorrupted firmware.
4. **Detecting Election Mode**. The firmware can determine whether the system is in test mode (LAT) or not. This means malicious firmware can respond correctly to the pre-election testing and incorrectly to the voters on Election Day.
5. **Election Management System**. The testers were able to bypass the Sequoia WinEDS client controlling access to the election database, and access the database directly. They were able to execute system commands on the host computer with access only to the database. Further, the testers were able to exploit the use of the autorun feature to insert a malicious program onto the system running the Sequoia WinEDS client; this program would be able to detect the insertion of an election cartridge and configure it to launch the above attacks when inserted into an Edge.
6. **Presence of an Interpreter**. A shell-like scripting language interpreted by the Edge includes commands that set the protective counter, the machine's serial number, modify the firmware, and modify the audit trail.
7. **Forging materials**. Both the update cartridges and voter cards could be forged.

The report presents several scenarios in which these weaknesses could be exploited to affect the correct recording, reporting, and tallying of votes.

6.2 Diebold

The team investigating the Diebold system identified several issues. They fall into several classes:

1. **Election Management System**. The testers were able to penetrate the GEMS server system by exploiting vulnerabilities in the Windows operating system as delivered and installed by Diebold. Once this access was obtained, they were able to bypass the GEMS server to access the data directly. Further, the testers were able to take security-related actions that the GEMS server did not record in its audit logs. Finally, with this level of access, the testers were able to manipulate several components networked to the GEMS server, including loading wireless drivers onto the GEMS server that could then be used to access a wireless device plugged surreptitiously into the back of the GEMS server.

2. **Physical Security**. The testers were able to bypass the physical controls on the AccuVote Optical Scanner using ordinary objects. The attack caused the AV-OS unit to close the polls, meaning the machine could not tally ballots at the precinct or inform voters whether they had "over-voted" their ballot. Similarly, the testers were able to compromise the AccuVote TSx completely by bypassing the locks and other aspects of physical security using ordinary objects. They found an attack that will disable the printer used to produce the VVPAT in such a way that no reminders to check the printed record will be issued to voters.
3. **AccuVote TSx**. The testers found numerous ways to overwrite the firmware in the AccuVote TSx. These attacks could change vote totals, among other results. The testers were able to escalate privileges from those of a voter to those of a poll worker or central count administrator. This enabled them to reset an election, issue unauthorized voter cards, and close polls. No knowledge of the security keys was needed.
4. **Security Keys for Cryptography**. The testers discovered that a well-known static security key was used by default.

The report presents several scenarios in which these weaknesses could be exploited to affect the correct recording, reporting, and tallying of votes.

6.3 Hart

The team investigating the Hart system identified several issues. They fall into several classes:

1. **Election Management System**. The testers did not test the Windows systems on which the Hart election management software was installed because Hart does not configure the operating system or provide a default configuration. Hart software security settings provide a restricted, Hart-defined environment that the testers bypassed, allowing them to run the Hart software in a standard Windows environment. They also found an undisclosed account on the Hart software that an attacker who penetrated the host operating system could exploit to gain unauthorized access to the Hart election management database.
2. **eScan**. The testers were able to overwrite the eScan firmware. The team also accessed menus that should have been locked with passwords. Other attacks allowed the team to alter vote totals; these attacks used ordinary objects. The team, in cooperation with the source code review team, was able to issue administrative commands to the eScan.
3. **JBC**. The team developed a surreptitious device that caused the JBC to authorize access codes without poll worker intervention. The team verified that the mobile ballot box (MBB) card can be altered during an election. The team also found that post-election safeguards to prevent the altered data on a tampered MBB card from being counted can be easily bypassed.
4. **eSlate**. The testers were able to remotely capture the audio from a voting session on an eSlate with audio enabled, thereby providing an attack that violates voter privacy. The team was also able to force an eSlate to produce multiple barcodes after printing

"BALLOT ACCEPTED" on the VVPAT records. This could cause a county that used bar code readers to read the VVPAT to produce erroneous vote totals.

The report presents several scenarios in which these weaknesses could be exploited to affect the correct recording, reporting, and tallying of votes.

6.4 Discussion

The red teams demonstrated that the security mechanisms provided for all systems analyzed were inadequate to ensure accuracy and integrity of the election results and of the systems that provide those results.

Electronic voting systems are critical to the successful conduct of elections in those jurisdictions where they are used. Given the importance of voting and elections in the governing of the State of California, one may safely say that these systems are "mission critical". Such systems need to be of the highest assurance in order to ensure they perform as required. Techniques for developing such systems are well known[7] but, sadly, not widely used. Vendors would do well to adopt them for electronic voting systems.

Similarly, many components of voting systems run on commercial operating systems. A non-secure underlying operating system offers attackers avenues into the software that the operating system runs, in this case the vendors' election management systems. Hence vendors must ensure that whatever underlying operating system their software runs on meets the security requirements that their software meets.

A key idea underlying high assurance techniques is that *security should be part of the design and implementation of the system and not added on "after the fact"*. The reasons for this need not be repeated here. Many of the components tested appear to have been hardened by taking their basic design and adding security features. As a result, the testers were able to exploit inconsistencies between the protective mechanisms and that which they were intended to protect.

Vendors should assume the components of the voting system will be used in untrusted environments in which they cannot be adequately monitored. Thus, their physical protections should be "hardened" to withstand determined attack. The added barrier that such mechanisms create will hamper the ability of attackers to obtain illicit access to the components even if lapses in procedural mechanisms allow them unobserved or unfettered access to the systems.

Of equal importance is the ability to detect when such attacks occur. Again, this speaks to security mechanisms as being "layered"; one must implement mechanisms to prevent compromise, and then add mechanisms (which may be the same as the previous ones) to enable observers to detect compromise should the preventative mechanisms fail.

[7] See for example Elisabeth Sullivan's excellent discussion in [9], chapters 18 and 19.

Because detection requires that people take some action, the security mechanisms require that specific procedures be designed in order to ensure that failure of the preventative mechanisms, and success of the detection mechanisms, are properly handled. An excellent example comes from the realm of physical security. A common belief is that tamperproof tape is sufficient to detect the violation of preventative mechanisms; for example, sealing a bay with tamperproof tape enables one to detect that the bay has been opened. Two problems arise. First, there must be a procedure to check the tamperproof tape. Second, an attacker can often acquire the same tape as is used to protect the systems. The attacker simply removes the tape showing evidence of the tampering, and replaces it with her own tape. Unless the original tamperproof tape has unique serial numbers and the observers check those serial numbers, the detection mechanism is defeated. Unless the customers follow an appropriate procedure (here, checking that the tape is intact and the intact tape has the right serial numbers), the security mechanism is easily defeated.

Finally, the red teams wish again to emphasize the inadequacy of "security through obscurity" as a key defensive mechanism. No security mechanism should ever depend on secrecy. At best, secrecy should be a single security mechanism in a layer of defensive security mechanisms. In this study, when vendors failed to provide software that would have helped the red teams expedite the testing process, the failure became a *motivation* for the red teams to construct equivalent software to carry out the attacks. The only thing lost was time that could have been used for testing. Given the constraints under which the red teams operated, a well-financed team of attackers, with plenty of time to plan attacks between elections, could do considerably better.

7.0 Conclusion

Neither this report nor the individual system public reports count the successful, unsuccessful, and untried attacks. The reason for this is that a comparison of the systems based on raw counts from this study is at best meaningless and at worse misleading. First, judging the vulnerability of a system requires understanding both the nature and the implementation of the policies and procedures under which it is used. A system that has 10 vulnerabilities that can be remediated by proper, realistic procedures can meet a set of requirements better than a system with only one vulnerability that cannot be remediated by realistic procedures. As the red teams ignored compensating controls and mitigations, the raw counts of successful, unsuccessful, and untried attacks do not indicate which would still be successful in the face of compensating controls—and how realistic those compensating controls would be.

Nor does this report characterize the difficulty, or lack thereof, of carrying out the successful attacks. This question has two parts. The first is how difficult it was to develop the attack mechanisms and (when needed) software. The second is how difficult it would be to carry out the actual attack, given the mechanisms and software developed. Consider an attack that replaces the firmware of a voting system with firmware that is malicious. Developing the malicious firmware, and building the software mechanism to install it, requires an expert or team of experts. But carrying out the attack requires only access to a voting system (i.e., someone voting) and not technical expertise. Further, the precise

procedural controls in place affect the difficulty of both phases of the attack, and the red teams focused only on the technical aspects of the systems.

As mentioned earlier, the red teams encountered difficulties in acquiring information and tools that would have allowed the study to go faster, and explore more potential threats and attacks. For the future, we suggest the Secretary of State adopt regulations to make the delivery of documents, software, and other material mandatory *before* certification. Then from the beginning of the review, the testers and reviewers will have all material at hand.

Perhaps wording similar to the following would accomplish this goal:

> "Source code materials shall be submitted to an approved escrow company for placement in the escrow facility. The contents of source code materials shall be in a form, and include the source code, tools and documentation, to allow the complete and successful compilation and installation of a system in its production/operational environment with confirmation by a verification test by qualified personnel using only this content."

The intent is that the vendor should provide everything necessary for the testers to build the system from the escrowed materials, and that those materials include the source code. This would have, for example, required Hart to escrow the uploading program that the red and source code review teams reconstructed, and allowed the teams more time to test potential attacks and problems using that tool.

8.0 Acknowledgments

Many people assisted us on this study. We wish to thank Jason Heyes, Ryan Macias, and Miguel Castillo of the Office of the California Secretary of State for putting up with the odd hours that we worked, and taking care of the electronic voting systems; Chris Maio for helping us set up some equipment; Debbie O'Donoghue and Lowell Finley of the Office of the California Secretary of State for administrative support and, especially, for helping us communicate with the vendors; and all the members of the accessibility review, source code review, and document review teams. Throughout this project, the other teams provided information, suggested possible attacks, helped us analyze results, and in some cases came up to Sacramento to help us analyze systems and try different attacks. Finally, we thank David Wagner for his critical support throughout this project. This was truly a group effort, and it is a pleasure to acknowledge all involved.

9.0 References

[1] *Master Agreement 06158101 between the Secretary of State and the Regents of the University of California*, May 4, 2007

[2] *COBIT 4.1: Framework, Control Objectives, Management Guidelines, Maturity Models*, IT Governance Institute, Rolling Meadows, IL (2007).

[3] *ISO 9001:2000, Quality Management Systems—Requirements*, International Standards Organization, Geneva, Switzerland (2000).

[4] *Capability Maturity Model® Integration (CMMI^{SM})*, Version 1.1, Technical Report CMU/SEI-2002-TR-012, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA 15213 (Mar. 2002).
[5] Office of Government Commerce, *Managing Successful Projects with PRINCE2*, The Stationary Office, London, UK (May 2005).
[6] *A Guide to the Project Management Body of Knowledge (PMBOK® Guide)*, Third Edition, Project Management Institute, Newtown Square, PA (2004)
[7] *Minimum Security Requirements for Federal Information and Information Systems*, FIPS PUB 200, Computer Security Division, Information Technology Laboratory, National Institute of Standards and Technology, Gaithersburg, MD 20899 (Mar. 2006).
[8] "Government Agencies Posting Sensitive 'Need to Know' Material Online", Fox News (July 12, 2007); available at http://www.foxnews.com/story/0,2933,289011,00.html.
[9] M. Bishop, *Computer Security: Art and Science*, Addison-Wesley Professional, Boston, MA (2003).
[10] "Voting System Standards", Federal Election Commission, Washington DC (2002); available at http://www.eac.gov/election_resources/vss.html.
[11] *Approval of Use of Diebold Election Systems, Inc. DRE & Optical Scan Voting System*, Office of the California Secretary of State, Sacramento, CA (2006); available at http://sos.ca.gov/elections/voting_systems/diebold_cert.pdf.
[12] *Approval of Use of Sequoia Voting Systems, Inc. DRE & Optical Scan Voting System*, Office of the California Secretary of State, Sacramento, CA (2006); available at http://sos.ca.gov/elections/voting_systems/sequoia_cert.pdf.
[13] *Approval of Use of Hart InterCivic System 6.2.1 DRE & Optical Scan Voting System*, Office of the California Secretary of State, Sacramento, CA (2006); available at http://sos.ca.gov/elections/voting_systems/2006-09-22_System_6_2_1.pdf.
[14] E. Barr, M. Bishop, and M. Gondree, "Fixing Federal E-Voting Standards," *Communications of the ACM* **50**(3) pp. 19–24 (Mar. 2007).
[15] *Voluntary Voting System Guidelines*, Election Assistance Commission, Washington DC (2005); available at http://www.eac.gov/vvsg_intro.htm.
[16] "Citing Problems, U.S. Bars Lab from Testing Electronic Voting", *New York Times* p. 1 (Jan. 4, 2007).
[17] D. Kawamoto, "SEC Filing Acknowledges 'Pretexting' in HP Board Probe," *CNET News*.com (Sep. 6, 2006); available at http://news.com.com/SEC+filing+acknowledges+pretexting+in+HP+board+probe/2100-1014_3-6112710.html.
[18] M. Bishop, *Protocol for Red Team Testing*. Available at http://www.sos.ca.gov/elections/voting_systems/ttbr/red_team_protocol.pdf.

Executive Summary

The California Secretary of State entered into a contract with the University of California to test the security of three electronic voting systems as part of her Top to Bottom Review. Each Red Team was to try to compromise the accuracy, security, and integrity of the voting systems without making assumptions about compensating controls or procedural mitigation measures that the vendor, the Secretary of State, or individual counties may have adopted. The Red Teams demonstrated that, under these conditions, the technology and security of all three systems could be compromised.

This report presents the findings of the Red Team testing on the Diebold Election Systems Incorporated voting system (Diebold GEMS 1.18.24/AccuVote), as performed by the following team members: Robert P. Abbott (team leader), Mark Davis, Joseph Edmonds, Luke Florer, Elliot Proebstel, Brian Porter, Sujeet Shenoi, and Jacob Stauffer.

The Red Team tested the physical and technological security of the hardware and software included in the Diebold voting system in order to identify vulnerabilities that could be exploited to violate the accuracy, secrecy, or availability of the systems and their auditing mechanisms. Red Team testing began on June 14 and concluded on July 19, during which time the team was testing both the Diebold Election Systems Incorporated voting system and the Hart InterCivic voting system[1]. This limited time frame did not allow the team to fully test the systems. Thus, results from this study should not be viewed as a complete report on all of the vulnerabilities that may exist in this system.

As tested, the Red Team found vulnerabilities in the Diebold GEMS 1.18.24/AccuVote system, which – in the absence of procedural mitigation strategies – could be exploited to compromise the accuracy, secrecy, and availability of the voting systems and their auditing mechanisms.

I. Introduction

The Red Team undertook the task of attempting to violate the physical and technological security measures of the Diebold Election Systems Incorporated voting system (Diebold GEMS 1.18.24/AccuVote) in order to discover exploits that have the potential of violating the accuracy, secrecy, or availability of voting systems and their respective auditing mechanisms. This analysis was performed by the following team members: Robert P. Abbott (team leader), Mark Davis, Joseph Edmonds, Luke Florer, Elliot Proebstel, Brian Porter, Sujeet Shenoi, and Jacob Stauffer.

In developing our attacks, we made no assumptions about constraints on the attackers. "Security through obscurity" – or the practice of assuming a veneer of security by relying on attackers not having access to protocol specifications or of using tools that are perceived to be difficult to acquire – is not an acceptable option for any system that can't afford to have its security compromised. Our study examined what a dedicated attacker could accomplish with all possible kinds of access.

[1]The findings from the Hart system are presented in a separate report.

We present our findings here. In Section 2, we present an overview of the Diebold voting system and how the system components interact. Sections 3 and 4 offer a more detailed overview of the vulnerabilities we exploited and what we believe, based on our research, are some viable attack scenarios, although not all of these scenarios were tested. Finally, Section 5 presents some concluding remarks.

We also note here that there are a great number of details that are not present in this public report. In particular, we have taken great care to ensure that we are offering the maximum amount of detail without violating our non-disclosure agreements with the vendors and without providing a "road map" to would-be attackers. Though state and county procedures may mitigate the impact of potential attacks, we believe it is in the public's best interest that this report not provide too much detail To this end, we note that there are occasional references to previous studies throughout this report. In order to perform due diligence, we attempted to verify previous security-related findings, where applicable, on all of the devices we were testing. Because citing those reports specifically in this report would provide the road maps we are seeking to avoid disclosing, all reference citations have been redacted to the confidential report.

II. Device Descriptions

The following is a full list of Diebold GEMS 1.18.24/AccuVote devices evaluated during the Top to Bottom Review. This section will give a brief description of each device in the Diebold e-voting system outlining their functionality. Also included in this section is a full connectivity diagram, outlining all physical connections between devices, and full pictures of each device.

Components

1. **GEMS Server** - Diebold election management system software is called GEMS (Global Election Management Systems). It is run on a server that is manually configured by Diebold technicians. We use the phrase "GEMS server" to reference the entire physical server in its delivered configuration, including the operating system (Windows 2000 Server), all software – including but not limited to GEMS software, and the physical configuration of the server.

   The GEMS server is used to set up the ballot definition, create security and administrative smart cards, program TSx and AV-OS memory cards, and print paper ballots. During the election, the GEMS server is responsible for performing image processing on the ballots scanned by the Central Count AV-OS. After the election, the GEMS server tallies the election results and is used for generating election result reports and databases.

2. **AV-TSx** – The AV-TSx (AccuVote-TSx – also referenced throughout this document simply as TSx) is the DRE (Direct-Recording Electronic) voting terminal on which voters cast ballots. Diebold technicians informed us that a TSx is used at Election Central to program PCMCIA cards for the election; cards programmed in a TSx unit may be used in the unit in which they were programmed or in other TSx units. The TSx can perform initial setup (card programming) and final reporting to the GEMS server by memory card transfer, Ethernet connection, or modem line. Though other means exist for writing to and reading from the memory cards, the Red Team tested the configuration described by Diebold technicians.

3. **AV-OS** - The AV-OS (AccuVote Optical Scan) is an optical ballot scanner. The AV-OS uses an Espon 40-Pin memory card (or compatible card – though Epson discontinued production of these cards in 1998) to store configurations and election definitions. Diebold technicians informed us that cards may be programmed in the AV-OS in which they will be used during the election, or in another AV-OS unit. The AV-OS can perform initial setup (card programming) and final reporting to the GEMS server by memory card transfer, Ethernet connection, or modem line. Though other means exist for writing to and reading from the memory cards, the Red Team tested the configured described by Diebold technicians.

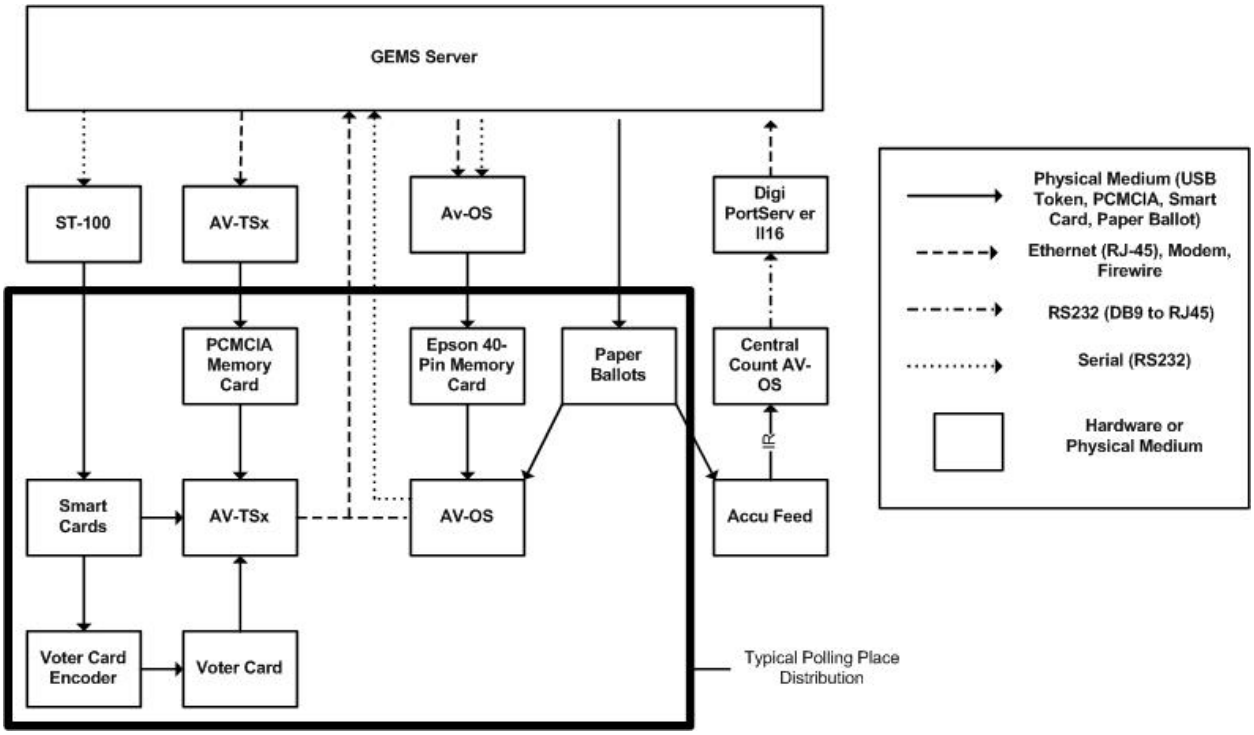
4. **Central Count AV-OS** – This AV-OS is connected to the AccuFeed to read paper ballots in bulk at a central count facility. It communicates to the GEMS server via the DigiPort Server II16, and the GEMS server performs all image processing of the paper ballots.

5. **AccuFeed** – The AccuFeed is used in a central count facility to feed paper ballots (cast at the polling places or by absentee voters) into the Central Count AV-OS. The AccuFeed communicates with the Central Count AV-OS via infrared.

6. **Smart Cards** – Smart cards are used to control the security and administration of an election. There are four distinct types of smart cards: Security Key Cards, Central Administrator Cards, Supervisor Cards, and Voter Access Cards. The first two are intended to be used only by Central Count officials, while Supervisor Cards are distributed to poll workers for their use during and after the election; the Central Administrator and Supervisor Cards grant access to the respective administrative menus. Voter Access Cards are used by voters as tokens which authorize each voter to cast a single ballot at a TSx unit. Smart cards are written either by the ST-100 card reader/writer connected to the GEMS server or (only in the case of Voter Access Cards) the Voter Card Encoder at the polling place.

7. **ST-100** – The ST-100 smart card reader/writer is connected to the GEMS server via a serial cable. It is used to encode the various smart cards used throughout the election process.

8. **DigiPort Server II16** – The DigiPort Server II16 is an intelligent network hub. It translates serial communication into Ethernet (and vice versa) in order to facilitate communication between the Central Count AV-OS units and the GEMS server.



Interactions Between Components

The GEMS server is used to create and define all aspects of an election, create security and administrative smart cards, and upload the election definitions to other components used during the election. The GEMS server encodes smart cards to be used by central count and polling place personnel using the ST-100.

The GEMS server is connected to both the TSx and AV-OS and the election definition will be downloaded to the removable memory cards for both systems: a PCMCIA for the TSx and an

Epson 40-Pin memory for the AV-OS. The GEMS server can also be used to create paper ballots for use in the election.

The TSx and AV-OS units used to create the original memory cards can be deployed to the polling place, or the cards can be deployed in other TSx and AV-OS units.

The smart cards programmed by the GEMS server through the ST-100 are used to program the Voter Card Encoder, to access administrative functions on the TSx units, to start and end the election on the TSx units, and (if applicable) to accumulate results from the PCMCIA cards used in other TSx units.

Each voter is given a Voter Card created by the Voter Card Encoder. The voter inserts her Voter Card into the TSx, and this authorizes her to cast a ballot on the TSx. Alternatively, the voter may receive a paper ballot that will either be read by the AV-OS at the voting site or accumulated and read by a Central Count AV-OS. If paper ballots are read at the central count site, they are fed into the Accu Feed device, controlled by an AV-OS and processed by the GEMS server.

At the end of the election day, the TSx and AV-OS units can transmit results to the GEMS server by a number of methods, including Ethernet or modem. The PCMCIA and Espon 40-Pin memory cards (from the TSx units and the AV-OS units, respectively) can be returned to central count where they are loaded into designated TSx and AV-OS units, which are connected, respectively, via Ethernet and serial connections. Alternately, the TSx and AV-OS units used in the polling stations can upload results via direct Ethernet and serial connections or via modem transmission[2]. The GEMS server tallies election results and prepares end of election databases and reports.

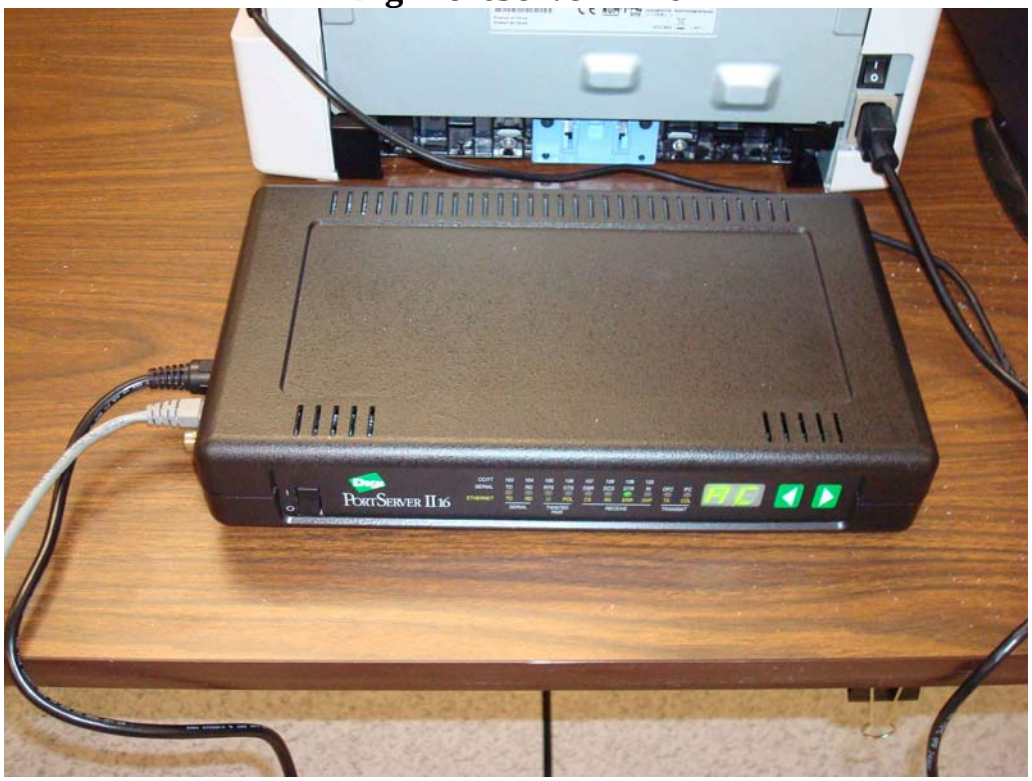
[2] Again, there exist other means for reading from the memory cards, but this is what Diebold technicians described to the Red Team as standard practice.

GEMS Server

Accuvote TSx

AccuVote OS (Optical Scan)

Digi PortServer II 16



SecureTech ST-100 PCMCIA Card Reader

Spyrus Voter Card Encoder and Smart Cards

AVOS and AVTSx Memory Media

III. Relevant Findings

In this section, we present a high-level description of the vulnerabilities we found in the Diebold voting systems. Our study was constrained by the short time allowed. ***The vulnerabilities identified in this report should be regarded as a minimal set of vulnerabilities***. We have pursued the attack vectors that seemed most likely to be successful. Other attack vectors not described here may also be successful and worth pursuing. This work should be seen as a first step in the ongoing examination of the systems, All members of the team strongly believe that more remains to be done in this field—and, more specifically, on these systems.

The systems and software versions we tested were:

Diebold GEMS 1.18.24/AccuVote

1. GEMS software, version 1.18.24
2. AccuVote-TSX with AccuView Printer Module and Ballot Station firmware version 4.6.4
3. AccuVote-OS (Model D) with firmware version 1.96.6
4. AccuVote-OS Central Count with firmware version 2.0.12
5. AccuFeed
6. Vote Card Encoder, version 1.3.2
7. Key Card Tool software, version 4.6.1
8. VC Programmer software, version 4.6.1

1. GEMS Server Vulnerabilities

There were stark discrepancies between the GEMS server as Diebold technicians delivered it and the GEMS server configuration as described in the Diebold documentation. The Diebold technicians assured us that the configuration we were given at the outset of the study was identical to the configuration Diebold technicians would supply to their customers (i.e. county officials). Thus, our findings are based on a system configured by Diebold technicians exactly as they informed us they would prepare a system for delivery.

The GEMS server is on a local area network (LAN) with other Diebold components, and this LAN is supposed to be isolated. However, even Diebold documentation reports that this requirement is not always met. Therefore, attacks via Ethernet against the GEMS server could reasonably be executed by personnel with physical access to the networking components (hubs/switches) in the isolated LAN or— if the Diebold LAN were intentionally or unintentionally connected to a public internet connection—by remote attackers

a. Windows Vulnerabilities

The Red Team performed vulnerability scans against the GEMS server. The results identified multiple vulnerabilities; primarily, these vulnerabilities existed because the Windows 2000 server (configured by the Diebold technicians) was not properly patched.[3] After noting these vulnerabilities, the Red Team was able to download an exploit from a

[3] Even if the Red Team had been expected to make other system configuration changes in order to make the GEMS server consistent with Diebold configuration documents, it would have been highly unreasonable for Diebold to expect the Red Team to patch Windows 2000 Server.

free public repository of well-known and documented exploits. This exploit gave the Red Team access of a Windows Administrator on the GEMS server.

Additionally, the Red Team noted that most standard Windows logging capabilities were either disabled or enabled in very limited states in the configuration provided by Diebold. This means that most malicious actions taken by attackers would not be traceable. More detail on the auditing configuration of this system is available in the report prepared by the 2007 TTBR Diebold Documentation Review Team.

Finally, the Red Team uncovered evidence that Diebold technicians created a remotely-accessible Windows account that, by default configuration (according to the Diebold documentation), can be accessed without the need to supply a password. There is evidence to suggest that this account is intended to be used by TSx units for dial-in access at the close of polls on Election Day, but the documentation for election officials never mentions this particular account by name. An attentive system administrator would notice the account. However, the responsibility should not be on election officials to discover remotely-accessible Windows accounts and act appropriately to ensure those accounts are not inappropriately accessed. Devices, as delivered to customers, should only have accounts that are well-documented and remote access that is necessary for the needs of the particular county. Undocumented remotely-accessible logins are contrary to generally-accepted security practices.

b. GEMS Databases
The Red Team used Windows Administrator access on the GEMS server to manipulate and corrupt GEMS databases. These actions could result in manipulated vote totals or in the inability to read previously-generated ballot definitions if no valid database backups were available (whether because the backups were not made or because the backups had also been corrupted). On election night, the inability to read results from the deployed TSx and AV-OS devices could render an election impossible to complete electronically. In this case, a hand count of paper ballots and VVPAT records would be the only option for deducing the intent of the voters who turned out on Election Day.

c. GEMS Audit Logs
The Red Team found methods for executing actions from within the GEMS server that could not be tracked by the GEMS audit logs, allowing malicious GEMS users to conceal actions they had taken while logged in. Additionally, the Red Team noted that one of the standard functions offered by GEMS is the ability for a GEMS administrative user to change the username of her account. This is a non-standard computing practice, and it could potentially be used by a rogue administrator to implicate another GEMS user (i.e. other elections personnel).

d. GEMS Election Configurations
The Red Team identified format string vulnerabilities that, when exploited, caused an election to run smoothly on a TSx unit until a voter from a particular precinct attempted to cast a ballot. When a voter from the affected precinct tried to cast a ballot on a TSx, the printer would generate an error, and the voter's ballot would be canceled. The voter

is notified about the error via a series of error messages that would be incomprehensible to the average voter, followed by this notification: "Your ballot has been canceled."

2. GEMS Server Networking Components

Using information gained from access obtained as the Windows Administrator user, the Red Team was able to guess the authentication credentials for the networking hardware supplied by Diebold, and gain root access on these devices. These root accesses would provide sufficient access for an attacker to manipulate every setting on the networking devices and on the server. Additionally, the Red Team was able to use this access on the GEMS server to install the drivers for a USB wireless dongle. This small device was then planted on the back of the server, ensuring remote access to the GEMS server even if it were disconnected from the Ethernet connection previously used to exploit the server.

3. Precinct Count AV-OS

The Red Team was able to verify the findings of some previous studies on the AV-OS unit; the impact of these was to alter vote totals in order to change the vote results on that machine.

The Red Team was also able to craft low-tech attacks that could cause an AV-OS unit to stop reading ballots, rendering it unusable for the remainder of the Election Day. This would not preclude voters from casting ballots or ultimately prevent ballots from being tallied. However, it could preclude ballots from being scanned at the precinct to help a voter determine if they "over-voted" their ballot.

4. TSx

a. TSx: Physical Security

The Red Team was able to violate the physical security of every aspect of the TSx unit, using only tools that could be found in a typical office. This guaranteed the access necessary to execute physical and electronic attacks. The team was also able to jam the locks, which would not only provide evidence of election tampering (the effects of which are unclear and would depend on county procedures) but which could also potentially render devices inoperable for future elections, let alone for the retrieval of election data already loaded on the device at the time of attack.

b. TSx: Malware

The team verified previous findings regarding multiple avenues for overwriting system firmware and software as well as for the introduction of malware that would affect the current software. These avenues, when exploited, are a platform for altering vote totals to potentially change the outcome of an election. They could also be leveraged to violate voter privacy[4] or enact a denial of service on affected devices.

Of potentially greater concern, the introduction of malware into a TSx unit could spread virally into the GEMS server via format string errors in the GEMS software as identified

[4] Cast ballots are stored – with timestamps – on the TSx in the order in which they were cast. For more details on this, please see the 2007 TTBR Diebold Source Code Team report.

by the team. TSx units use PCMCIA cards to store and transport election definitions and vote totals. When those vote totals are communicated back to the GEMS server (either by physical transfer of the PCMCIA card into a TSx unit connected directly to the server's LAN or over a dial-in connection), an exploited TSx could virally infect the GEMS server. Future TSx and AV-OS units connected to the GEMS server could likewise be infected as ballot definition files are transferred via serial or Ethernet connection.

c. TSx: Escalation of Privileges
The Red Team identified attacks that can escalate the privileges of a voter to those of a poll worker or those of a central count administrator. These attacks use tools that can be found in a typical office and could be executed by a very low-skilled attacker.

The privilege escalations can allow a voter to reset an election (deleting all electronic records of ballots cast so far on the system, including backup records), issue unauthorized Voter Access Cards (the single-use tokens used by voters to allow authorized voters to each cast only a single ballot at a TSx station), program a TSx unit to remotely call an attacker's modem, gather sufficient login information to gain unauthorized remote access to the GEMS PC, or close polls and view all ballots already cast on the particular TSx unit—all without any insider knowledge of election-specific data such as the security keys in use.

d. TSx: Default Static Key
The Red Team verified that a previously-identified default static key is still in use on the systems. Diebold documentation urges election officials to not use this static key and to generate their own election-specific keys. If election officials opt to use the static keys (or forget to change them), the TSx units display a particular icon on the screen to warn that the keys in use are insecure. A knowledgeable attacker could observe this icon and use the information being leaked by the TSx unit to craft more specific attacks for the system.

e. TSx: Malicious Voter Input
Multiple voter-accessible input fields on the TSx are susceptible to malicious input. The Red Team tested these input fields and observed erratic TSx behavior in response to proof of concept code. The team did not have time to study these vulnerabilities sufficiently to craft working exploits but notes that the lack of input validation allowed us to generate unusual printed output and/or crash the units when we provided malicious input to these fields.[5]

f. TSx: VVPAT
The Red Team found a simple attack that can put the Voter Verifiable Paper Audit Trail (VVPAT) printer out of service until the TSx unit is rebooted. This attack requires only tools that can be found in a typical office. Voters who were not aware that they should expect a printed version of their ballot for review would not observe anything unusual,

[5] Similar vulnerabilities exist in input fields accessible through administrative menus.

because one result of this attack is to cause the TSx to stop issuing reminders to voters that they should verify the printed record of their selections.

The team also found that the design of the TSx printer enabled us to devise attacks on the printed records that could covertly destroy VVPAT records using a common household substance.[6] This is particularly notable because the attack meets the following conditions:

1. It affects records printed before the attack is executed.
2. It affects records printed after the attack is executed.
3. It does not affect the way records are displayed to voters as they are produced – so as to avoid raising voter suspicion before the close of polls.
4. It does not affect the printer mechanisms or jam the printer – again, to avoid raising suspicion.

The impact of these attacks is to make many of the VVPAT-printed ballots completely unreadable and most of them barely or only partially readable. A successful attack could not only destroy records already printed by the VVPAT at the time of the attack but could also potentially destroy all records produced throughout the rest of the day by that particular VVPAT. The impact (once discovered) is highly visible, but when combined with an electronic attack that destroyed ballots, it could serve to effectively nullify most – if not all – of the ballots cast on a particular TSx unit. This attack is particularly viable on the TSx because the design of the VVPAT printer and the security casing for printed records allows the attack substance to linger undetected inside the machine until the end of election day; neither subsequent voters nor poll workers would know the attack had taken place until the printed records were removed at the end of Election Day.

g. TSx: PCMCIA card

The Red Team verified the results of other studies, which found that modifications to the contents of the PCMCIA card could affect the accuracy of vote totals. Additionally, the Red Team discovered that an attacker could extract GEMS server login credentials from the PCMCIA card in a TSx which had been programmed to dial in to GEMS at the close of polls.

[6] The Red Team has confirmed that the design of the printers on the Hart and Sequoia systems tested for the 2007 TTBR would not inherently enable attacks of the same magnitude. That is, no known attacks for the Hart or Sequoia printers could meet all four conditions listed above.

IV. Successful Attack Scenarios

The following attack scenarios were successfully carried out in the laboratory environment of the Secretary of State's testing facility.

1. Attack Scenario 1

   In this scenario, an attacker approaches a TSx unit during an election. The attacker may be a registered voter who first checks in with poll workers at the front desk and is issued a Voter Access Card, or she may simply walk into a busy polling station and slip unnoticed into the crowds and approach a TSx. She brings with her a few small tools for executing the attack; these tools, which can be concealed in the palm of her hand, are standard office equipment and would not be cause for suspicion even if they were seen by others in the polling station.

   In the span of time it takes for many voters to cast a typical ballot on a TSx unit, the attacker brings the system into an administrative mode, deletes the ballots cast thus far on the TSx (both the primary and backup records), and restarts the election on this particular unit. This effectively erases all electronic records of ballots already cast on this unit.

   Finally, the attacker executes a covert chemical attack that destroys or critically damages most, if not all, of the already established VVPAT records on this unit. The chemical attack also damages or destroys future records printed on this VVPAT, and it is highly unlikely that the attack will be observed until the close of election. This renders the paper trail almost completely unrecoverable, making it unlikely that the electronic attack would be detected, allowing the attacker to successfully nullify all ballots cast before the attack was launched.

2. Attack Scenario 2

   This scenario is a variation on the first, but in this scenario the attacker uses the "View Cast Ballots" option in the Polls Closed menu to view and print extra VVPAT records of ballots the attacker considers favorable. After printing an arbitrary number of favorable ballots, the attacker resets the election and leaves the TSx unit in Election Mode with zero recorded electronic ballots. At the close of polls, neither the VVPAT records nor the electronic records will be accurate.

3. Attack Scenario 3

   In this scenario, the attacker brings a stack of smart cards that she has formatted in such a way that a TSx unit will recognize them as generic Voter Access Cards. They do not contain election-specific credentials and thus require no insider information to prepare.

   He launches an attack that escalates his privileges to access a Poll Worker Menu and then uses the TSx to manually authorize the smart cards he brought. He can either use these himself to cast an arbitrary number of ballots or distribute them to collaborators who can each cast an arbitrary number of ballots. This effectively constitutes an electronic ballot box stuffing attack. While such an attack would likely be detected during the standard vote reconciliation process, it is not clear what would happen in the event more votes were recorded than there were voters who had signed in at the polling place.

4. Attack Scenario 4

In this scenario, the attacker launches a low-tech attack that can be discreetly executed at a Precinct Count AV-OS under the watch of a moderately attentive poll worker. The tools for completing the attack are small and easily concealed, and they can be obtained in a typical office. After the attack is completed, the AV-OS will no longer accept ballots for counting. While this won't preclude voters from casting ballots or ultimately having them counted, this attack does remove the AV-OS from service for the remainder of Election Day.

V. Potential Attack Scenarios

The team believes, based on our research, that the following scenarios would be successful; however, we didn't have the time necessary to successfully complete them.

1. Potential Attack Scenario 1

In this scenario, the attacker uses the TSx as a platform for gaining login credentials necessary to gain unauthorized remote access to the GEMS server. She executes an attack using tools found in a typical office to bring the TSx unit into a menu from which she can recover the phone number for the GEMS server's modem, as well as the username and length of password for a valid Windows account on the GEMS server.

She returns home with these credentials and remotely logs in to the GEMS server. With this Windows access, she is able to modify GEMS databases, altering vote totals. She can also simply delete GEMS databases, possibly making it impossible for election officials to read the electronic results from the TSx and AV-OS units at the end of Election Day.

2. Potential Attack Scenario 2

In this scenario, the attacker prepares a PCMCIA card with malicious software and brings it with him to the polling station. He bypasses the physical security protecting the PCMCIA card slots and inserts his card into the TSx unit. The malware on the card is loaded onto the TSx and overwrites the software on the system. He replaces the official election PCMCIA card, which is infected by the compromised TSx. When the card is later uploaded to the GEMS server, the malware is transferred to the GEMS server, which is used as a locus for spreading the infection to all other TSx units and future AV-OS memory cards. The attacker is able to use this viral infection to alter vote totals for the election at hand and to control future elections, as well.

VI. Conclusions

Although the Red Team did not have time to finish exploits for all of the vulnerabilities we discovered, nor to provide a complete evaluation of the Diebold GEMS 1.18.24/AccuVote system, we were able to discover attacks for the Diebold system that could compromise the accuracy, secrecy, and availability of the voting systems and their auditing mechanisms. That is, the Red Team has developed exploits that – absent procedural mitigation strategies – can alter vote totals, violate the privacy of individual voters, make systems unavailable, and delete audit trails.

Documentation Assessment of the Diebold Voting Systems

Candice Hoke[1] **Dave Kettyle**

Center for Election Integrity
Cleveland State University

July 20, 2007*

This report was prepared by the University of California, Berkeley under contract to the California Secretary of State as part of the "Top-to-Bottom" review of electronic voting systems certified for use in the State of California.

[1] Team Leader. Author affiliation is for identification only.
* Minor typographical and clarification changes were made after July 20, 2007, which are reflected in this document. One footnote was added and two were deleted.

Executive Summary

The California Secretary of State commissioned a comprehensive, independent evaluation of the electronic voting systems certified for use within the State. This team, working as part of the "Top to Bottom" Review ("TTBR"), evaluated the documentation supplied by Diebold Election System, Inc. Our analysis reached a number of conclusions, including:

- *Usability.* When the vendor's documentation is evaluated for its overall usability for local election officials' tasks, its deficiencies significantly outweigh its successes. While some topics and components are well-addressed, conscientious local election officials attempting to master the Diebold system will find the documentation presents numerous impediments to their managing the voting system correctly, in a manner that achieves high accuracy, security, and other core objectives.

- *Federal Testing Labs' Recommendation as Qualified.* The two testing laboratories that were contracted to evaluate the Diebold voting systems produced reports that differed greatly in their thoroughness, degree of detail, and in the adequacy of the bases for their qualification recommendations.

- *Wyle Report: Hardware and Firmware Review.* The reports submitted by Wyle Labs are reasonably thorough, providing details of examinations and testing that were conducted. The reports also analyze the test results, and evaluate these results in light of the federal 2002 voting system standards. The testing also appears to have followed standard methodologies for environmental and electrical testing of electronic components and supporting equipment,

- *CIBER Report: Software and Required Documentation.* The CIBER report on its evaluation of software and documentation pursuant to the 2002 federal standards is unusually brief (16 pages) given the complexity of the required evaluations. None of CIBER's reports on this system discuss in sufficient detail the methodologies, tests, or results that were obtained, and thus do not permit a reader to formulate an informed opinion on the degree to which the Diebold voting system met or exceeded the minimum federal standards for qualification.

- *Security Policy Differences.* Pursuant to the federal standards, Diebold submitted to CIBER a set of voting system security policies that it would mandate for localities purchasing the Diebold system. A comparative analysis shows that the security policies Diebold filed with CIBER were considerably more stringent and extensive than those it ultimately documented in Diebold's product manuals. These sharp differences raise the question of whether California counties are provided with adequate information to implement the security conditions under which the Diebold system was tested and approved.

- *Configuration Audit.* An audit comparing the California-certified configuration of the Diebold voting system with the configured system the vendor provided for the TTBR disclosed numerous

differences. A number of these configuration discrepancies involve an uncertified component, and unapproved and largely disabled security settings, raising serious questions about the voting system's accuracy, security, and reliability.

- *Security:* The vendor documentation misses opportunities to assist election officials who are striving to achieve secure elections. The vendor recommends certain security-oriented practices without an explanation of possible vulnerabilities. This approach tends to minimize serious security risks and sidestep mitigation strategies.

Table of Contents

Introduction

The California Secretary of State commissioned a comprehensive, independent evaluation of the electronic voting systems in use within the State. The project, also known as the "Top to Bottom" Review ("TTBR"), involved assigning four teams to each system with each focusing on a different aspect of the voting system. These included teams for source code review, accessibility, and security penetration (also known as the "Red Team") and documentation review. The teams worked under the overall supervision of the University of California. The Principal Investigators for the project, Professors Matthew Bishop and David A. Wagner, specialize in computer security. This document is the final report of the team that examined the Diebold voting system documentation the vendor submitted to support the system's certification reviews and ultimately, local election operations using the equipment.

The Diebold documentation review team was located at Cleveland State University and consisted of the two authors of this report plus an additional researcher.[2] We consulted often with the Source Code Review team located at Princeton University,[3] and the "Red Team,"[4] located at the University of California, Davis, and valued greatly these communications. This Diebold Document Review report should be read as a complement to the other two reports filed by the Diebold Source Code and Red Teams, but all of the conclusions offered in this report are solely those of its authors. The work started on May 31, 2007 and ended on July 20, 2007 with the delivery of this report.

Our report assesses the documentation for Diebold Election System, Inc.'s (Diebold) voting system (VS) approved for use in California counties. Materials the Secretary of State ("SOS") requested for this review included the confidential Technical Data Packages submitted to the Independent Testing Authorities (ITAs) for their qualification testing of the Diebold systems, the evaluative reports produced by the ITAs, and the documentation that is supplied to election jurisdictions in California who purchase Diebold systems. In reviewing the system, the Documentation Teams' assigned scope included the charge of assessing whether the federal testing laboratory reports supplied adequate documentation for

---

[2] The Diebold Documentation Review team's expertise includes technical areas (software engineering) plus analytic assessments of documents and legal standards. Election law professor Candice Hoke (J.D., Yale Law School; Director, Center for Election Integrity; Project Director for the Public Monitor of Cuyahoga Election Reform; Associate Professor of Law, Cleveland State University) served as team leader. David N. Kettyle and Thomas P. Ryan, both of whom served as technical-legal staff for the Public Monitor of Cuyahoga Election Reform, were the other two team members.

As Chapter 3 discusses, the Diebold Documentation Review team did not receive any of the vendor's technical documentation that it submitted as part of federal qualification testing until July 13, 2007. We chose to use the time between that date and the day this report was due (July 20th) to analyze and report on the almost 1000-pages of documentation rather allocate the scarce time remaining to including extensive citation to legal and other materials. For this background material citation, please consult the reports filed by the other two Documentation Teams (on Hart InterCivic and Sequoia voting systems.)

[3] The Diebold Source Code Team was comprised of David A. Wagner, (team leader), Joseph A. Calandrino, Ariel J. Feldman, J. Alex Halderman, Harlan Yu, William Zeller.

[4] The Diebold Red Team members were: Robert P. Abbott (team leader), Mark Davis, Joseph Edmonds, Luke Florer, Elliot Proebstel, Brian Porter, Sujeet Shenoi, and Jacob Stauffer.

their findings regarding the reviewed Voting System's adequacy under the voluntary federal standards.

Further, the SOS tasked independent documentation review teams to answer two additional major questions about the vendor's documentation taken as a whole. First, is the documentation ***complete***? This question directed us to determine whether the review team had received all the documents that the vendor would have supplied within the certification review process at the federal and State levels. Second, when considering the intended uses of the documentation by county, is the documentation ***sufficient***? Here, we considered whether, when consulted, it adequately supports the county Registrar of Voters' deployment of the VS in a manner that empowers the election officials to administer elections accurately, reliably, and with appropriate security and ballot privacy throughout the election processes. Adequate documentation will assist officials in avoiding errors that can cause systemic problems and alert them to strategies they can employ to prevent (or uncover) malicious tampering with the election equipment or vote totals.

1.1 Scope and Methodology

A voting system's documentation, like that of other technical systems, is normally intended to provide guidance for conducting normal operations and for diagnosing and correcting problems when unexpected situations occur. Where a complex system is at issue, such as the election management and voting device systems that are evaluated in this TTBR project, documentation usability by the intended audiences (for instance, election officials and poll workers) is especially important.

The SOS delineated the basic scope of this review in the Statement of Work (May 2007), with a functional orientation underlying the charge. The three Documentation Review teams (for the three voting systems under evaluation; Hart Intercivic and Sequoia are the other two vendors) then developed some initial methodological principles to guide our assessments and modified our template as needed throughout the review process.

An early focus was the 2002 Federal Election Commission Voluntary Voting System standards (hereafter *VSS*),[5] as it contains some criteria for assessing documentation. We distilled some analytic criteria from the VSS. Because of prior public assignments in Ohio, this Diebold Documentation Review team were able to draw on their prior reports and discussions with Ohio election officials who have been using largely the same Diebold voting system as was the subject of this California review. Our knowledge of some complex tasks that face local election officials, and of unexpected conditions that can arise and for which they need documentation assistance, led this Team to approach a great deal of the review tasks with a pragmatic approach. We began and ended this review motivated by the concern that the documentation provide a high quality of support for conscientious local officials as they go about their important duties.

Our team's work on the VSS analysis and documentation methodology, as well as on TTBR security policies and practices (instituting required physical and electronic security) began in May 2007 after the contracting had largely been completed. Our documentation review officially began in the first week of June, with all analysis scheduled to be completed except for drafting of this report by July 13th. But two of the long-sought Technical Data Packages (TDP) arrived on exactly July 13, 2007— well over one month after they should have been provided and on the exact date that the work scope agreement had stated evaluation work would end. Despite the late date, this Documentation Team proceeded to analyze the TDP documents relating to the most crucial aspects of the federal ITA qualification reviews as well as those that focused on issues that had arisen during the course of this study.

After the documentation arrived, our first task was to conduct an inventory to evaluate its ***completeness.*** As this report establishes, completeness was never achieved despite significant efforts.

5 http://www.eac.gov/election_resources/vss.html.

We used the ITA citation lists to determine that we had not been provided with any proper "TDP" documentation despite having received a CD that was marked "Diebold TDP." Most of our evaluation period was devoted to the operators' manuals, also known as user manuals, and the ITA reports.

On July 4th, the Diebold documentation team traveled to Sacramento to work in the secure "red team" testing room located in the SOS facility in Sacramento. This trip was designed to permit us to work with the Diebold VS in conjunction with the documentation. While we were not able to conduct a "walk-through" of an entire election preparation and tabulation using the manuals because of certain limitations, we were able to complete some practical tests of the documentation's adequacy. Additionally, we conducted a configuration audit while there. One of the most valuable aspects of the visit was that all team members present were able to discuss the TTBR project and preliminary findings. We also had an opportunity to meet with several California county election officials who visited the red team's testing room.

Throughout the review process, the Diebold Documentation Team provided support and consulted extensively with the other Diebold teams (Red/Penetration Team and Source Code Team) and with the other two voting systems' Documentation Teams.

Partly because of the paucity of technical documentation initially provided, this team's early focus was on the ***sufficiency*** of the documentation. We evaluated its sufficiency in two major respects: (1) whether the documentation supported the ***certification decisions*** that were made approving the system with regard to its accuracy, security, reliability and other criteria; and (2) whether the documentation array designed for counties purchasing the VS was organized and written in a manner that enabled ***local officials to manage and operate the voting system effectively***. The Secretary of State's charge to Documentation Review teams (exclusive of the accessibility team) resulted in our evaluating the documentation's usability and sufficiency for local operations according to core criteria or interrelated axes that included: (a) accuracy and reliability, (b) security; (c) ballot secrecy; (d) auditability, and (e) overall usability for the intended audiences. In general, we considered the local support documentation package as a whole instead of segregating particular manuals for evaluation.

1.2 Limitations

While the incompleteness of the documentation submitted for our review, and the extraordinary tardiness with which the Technical Data Packages were eventually produced, were significant impediments, some limitations of this study can be traced to its being the first of its kind, there being no methodological precedent to follow or improve upon. Future VS assessments will likely not experience the same limitations.

The major limitations on all three documentation review teams included:

- the lack of an opportunity to discuss with local election officials their experiences, and those of the voters in their counties, in using the VS under review;
- the unavailability of operational logs from the voting system that would enable an evaluation of normal and exceptional conditions, including tampering or system failures;
- the fact that the VS components were not observed in use by election staff and voters, in actual election preparation activities, and in voting, tabulation and reporting thereafter;
- time constraints that limited discussion and critique of other teams' reports, both within the Diebold teams and among all Documentation Review team members before the submission date;

- the inability to verify that the operational documentation that was provided to the review teams was the same that was provided to (1) the ITAs and state certification authorities, and (2) the California county election officials; and
- the unavailability of any incident reports and operational questions that have been submitted to the vendors by county election officials, and the responses of vendor personnel.

CHAPTER 2

System Overview

Note: The following overview of the Diebold voting systems that were reviewed for this report is geared toward the non-technical reader who is unfamiliar with election technology. As such, in places it sacrifices technical precision for general accessibility. Readers more familiar with the sorts of technical matters discussed in this report may wish to skip this chapter.

2.1 Voting System Components and Configuration

The Diebold voting system approved for use in California has a central "nerve center" software application named GEMS plus several types of voting devices that can be used by voters to cast ballots. Polling location voting device options include a (1) touchscreen unit named the TSx (usually deployed at Election Day polling locations or for early voting) and (2) optical scanners that read paper ballots, and (3) may be set up to have both voting device options. The TSx is required to include the printer for a paper audit trail so that voters can review and verify their ballot selections before casting their votes. The Diebold optical scan options are subdivided into (a) a "precinct count" system where voters can feed their paper ballot into the scanner that reads and tabulates their votes, and (b) a "central count" system set up to count hundreds or thousands of ballots at the county's offices. The "central count" scanners are linked together using a network that connects to the "GEMS server" – a computer on which the nerve center software is housed. GEMS tabulates the votes from both the touchscreen units and paper optically scanned ballots, and issues reports of voting results.

This section first provides the list of Diebold voting system components approved for California and then discusses them in greater technical and operational detail. California counties determined which of the certified system components would be purchased and deployed within their boundaries, given their resources and other factors.

The SOS provided to the TTBR researchers this listing of certified Diebold voting system components:

- GEMS, v.1.18.24
- AccuVote-TSX with AccuView Printer Module and Ballot Station f/w, v.4.6.4
- AccuVote-OS (Model D) with f/w v.1.96.6
- AccuVote-OS Central Count with f/w v.2.0.12
- AccuFeed
- Vote Card Encoder, v.1.3.2
- Key Card Tool Software, v.4.6.1
- VCProgrammer Software, v. 4.6.1

The "version number" follows the name of each component and designates the particular software

program or hardware component that has been approved for use. A component with the same name but a different version number is not legally certified or approved for use within the State. Similarly, any other software program or hardware component that interacts with these certified components is arguably prohibited from use in California unless it receives certification.

Importantly, under the federal Voting Systems Standards of 2002 (hereafter VSS), voting systems are tested, qualified and certified as *systems*, not merely as isolated components. This point is important because a deficiency or incompatibility in one part of a voting system can jeopardize the reliability, accuracy and security of other parts. Thus, when any component is revised or upgraded, with regard to its software (including embedded "firmware") or hardware, the VSS requires that the entire VS within which the component will, or could optionally function, be retested and re-certified. This approach ensures that the upgrade will not trigger unexpected but injurious interference with another portion of the VS.

2.2 GEMS Election Management Software

The GEMS software application is the "election management system" component. It is used in election preparation to create and configure ballots (both electronic and paper ballots), and programs memory cards for use in Diebold voting devices. Post-election, the main GEMS computer or "server" receives voting data from memory cards, tabulates, and then reports election results. GEMS sends or "uploads" configuration data for the AV-OS and AV-TSx units onto memory cards, which then store voting data from ballots cast in the election. These memory cards are then returned to the election offices for vote tallying after the polls close.

At the close of the election and after the memory cards are returned to the county offices, the voting data must be uploaded from the memory cards into the GEMS system for tabulation. This uploading typically occurs via networked devices of the type in which the voting data was stored. The TSx touchscreen memory cards are read by TSx devices linked to the GEMS PC by Ethernet. The memory cards from precinct-based optical scanners (AV-OS) must be inserted into the same type of scanner that recorded the voting data but this scanning unit is connected to the GEMS server. GEMS then deposits the vote data into its election database that was configured by county election employees prior to Election Day.

2.3 TSx Touchscreen Voting Device, VVPAT Printer, and Related Equipment

The Diebold Accuvote TSx is a touchscreen Direct Recording Electronic (DRE) voting machine that according to California law, is required to be equipped with a voter verified paper audit trail unit – otherwise known as a ***VVPAT printer***. The printer is designed to print a record of the voter's ballot choices, which then rolls up into a locked canister inside the TSx unit. The voter does not receive the paper record or a receipt of voting choices registered on the electronic unit but can view and verify the printed choices by looking through a small plastic window after each subset of ballot choices are printed.

The TSx unit is normally deployed to polling places. Its use requires a ***"smartcard***" or "voter access card" to initiate a ballot and authorize the TSx to record votes. A computer chip is embedded in the card. The ***Vote Card Encoder*** is a small hand-held unit slightly bigger than a credit card. Poll workers must be trained to use the encoder so that it will encode the smartcard to bring up the correct ballot (or accessibility functions and options, such as an audio ballot) when the voter inserts the smartcard into the TSx. An activated smartcard permits only one ballot to be cast. After a cast ballot has been recorded, the TSx deactivates the smartcard/voter access card so that it cannot be used to cast a second ballot. The card

then emerges, making a clicking sound when it automatically pops partially out from its slot. The voter is to return the smartcard to poll workers, who can then encode it so that it will be usable for subsequent voters.

When the voter (or poll worker) inserts a properly encoded smart card into the TSx, the TSX shows a touchscreen display or activates an audio ballot. As each ballot is cast, the AV-TSX stores an electronic record of the votes associated with that ballot onto a file on the memory card. When the polls have closed, the TSx counts all of the votes and can be directed to print out either a summary tape showing the combined vote tallies without breaking them down by precinct, or a longer tape that shows the tallies by precinct.

Before Election Day, the TSx units require substantial preparation. The GEMS computer is used to generate certain "ballot definitions" that are stored on the memory cards that will be in the TSx units. The ballot definitions store the races, candidate names, ballot issues, and other information in a legally regulated manner. The memory cards are created ("burned") in a process that has some TSx units in the county election offices linked to the GEMS computer; the memory cards receive the ballot definitions via the TSx.

The election officials are supposed to set up logic and accuracy testing for each TSx unit to ensure that when the memory card is inserted, the ballots can be brought up to the screen and will correctly register votes when the on-screen buttons are pushed. Then the memory cards are to be packed into sealed bags and marked for the correct polling place that will use the particular ballots that have been stored on the card, or are left in the TSx units and sealed for delivery to polling places.

The TSx machines are delivered to the polling location with the correct memory card already inserted and inside the locked compartment. Poll workers are instructed to run a ***"zero report***" to demonstrate that the ballot counters are registering zeros—indicating that no votes have been stored n the cards or machine. Poll workers typically are also charged with changing the printer paper as needed in the "AccuView Printer Module."

The TSx runs a Diebold-prepared version of the Microsoft Windows CE operating system. ***BallotStation*** is an additional application loaded on the firmware that provides the user interface that voters and poll workers see on the screen. BallotStation handles a number of crucial functions including managing the interaction with the voter, and accepting, recording, and counting votes.

The TSx memory cards must reach a location that permits voting data to be transferred ("uploaded") into the GEMS computer. Several different options are used in California. Sometimes poll workers are trained to remove the memory card from the machine's sealed slot and pack it for transport to county election headquarters (or to a regional transmission center) so that the electronic vote records can be transferred into the GEMS computer ("uploaded") for tabulation and election results reports. Alternatively, the county can specify that the TSx units are to be transported back to the county election offices with the memory cards undisturbed-- still sealed inside the TSx units. Then the election staff can record and break the seals, and remove the memory cards for uploading into GEMS.

Alternatively, a county might use the TSx internal modem to send vote data to GEMS over the regular phone lines. The TSx can be configured with the telephone number, username and password to connect to the GEMS server. In contrast to the AV-OS, the TSx software uses encryption and authentication software for such calls.

2.4 Optical Scanning at the Polling Place: Precinct Count AV-OS

One version of the Diebold optical scanner, the Accuvote OS (AV-OS), is a "precinct-count" optical scan machine. It is used to scan paper ballots at the polling place. Normally the voter will bring over the paper ballot that has been marked by filling in ovals beside ballot choices. Then, with the poll worker's assistance, the voter feeds the ballot into the scanner for reading and recording the votes. Thereafter, the scanner drops the scanned paper ballot into one of the compartments beneath the scanner, into a locked and sealed ballot box.

During an election, poll workers provide a paper ballot to a valid voter, who then marks the ovals with an approved pen or pencil., When the voter feeds the marked ballot into the AV-OS, the unit scans the ballot, interprets the marked votes, maintains a race counter to increase the vote counts as ballots are read, and then drops the ballot into the locked ballot box. If a voter filled in more than the allowed number of ovals in a given race or question, thus causing an "overvote," the AV-OS can return the ballot to the voter. The voter can then request a new ballot.

The approved California AV-OS "Model D" uses embedded software ("firmware") that is numbered as version 1.96.6. As with the TSx, the precinct count AV-OS memory card must be configured prior to the election with electronic ballot definitions that are created on the GEMS computer. Then, after testing it for accuracy and functionality, the memory card can be left in the slot on the front of the machine or can be packaged and delivered to the polling place for insertion on Election Day. The memory card ballot definition stores the names of races and candidates, plus the directions used for tallying and printing election results reports.

The county has several options for receiving the AV-OS voting data at the close of the election. At the close of the polling place, the unit can be returned to election headquarters with the memory card sealed inside or poll workers can remove the memory card and send it to be uploaded into the GEMS server. GEMS can then tabulate the results and issue the election reports. Or, the internal modem that is embedded in the AV-OS may be used to transmit vote data over the phone lines. If the modem option is used, the AV-OS apparently sends the vote data in the clear with no authentication or encryption.

2.5 Election Office Optical Scanning: Central Count Tabulation (AV-OS)

The AccuVote-OS (also known as the AV-OS) is the same hardware scanner that is used for the precinct count optical scanning but it has a different embedded software ("firmware") installed. Its configuration allows it to be linked with a number of other AV-OS units via a network whereby voting data can be sent into the GEMS server from many scanners concurrently scanning ballot batches. Firmware version 2.0.12 designates the machine is configured for 'central count" as opposed to "precinct count." Central count AV-OS is often used to count absentee ballots as well as provisional and damaged but "remade" paper ballots at county election headquarters or another centralized location.

Unlike the precinct count AV-OS, the AV-OS central count units' operation is largely controlled by GEMS. While the units scan ballots and interpret the ballot marks, the AV-OS central count uploads the voting data to GEMS and does not tabulate or keep any record of votes on the unit. The central count AV-OS memory card needs no ballot definitions and only has some technical information regarding the particular scanner so that it can be individually tracked as it scans ballots. It can be used with or without an automatic ballot feeder called the ***AccuFeed.***

2.6 Additional Certified Software Components

The ***Key Card Tool Software***, v.4.6.1, is used to set security keys for smartcards (voter access cards) and supervisor cards. It can also be used to set passwords for use with supervisor cards. It permits election officials to use a unique security key for each election to make it more difficult to tamper with election equipment by means of unauthorized access cards. A PC-based application, it can be installed on the GEMS computer.

The ***VCProgrammer Software***, v.4.6.1, can be used to program voter access cards to activate the proper ballot on TSx for a given voter. It differs from Voter Card Encoder, which is limited to programming up to 8 different ballots. VCProgrammer can retain and program an unlimited number of ballot definitions for an election, which makes it a better choice for creating voter access cards at early voting locations.

CHAPTER 3

Degree to Which Documentation Was Complete

3.1 Overview

The California Secretary of State specified the range of documentation the voting system (VS) vendors were to provide for the TTBR. The review was to encompass all Diebold technical documentation submitted as a part of the federal testing lab's evaluation as well as all VS operating manuals, among other items. This documentation was to include all manuals that guide the vendor's technicians in initial set up or other work as well as those that support local election officials in their operation of the systems.

The Diebold documentation review team never received all of the expected TTBR documentation, despite this team's follow-up with detailed lists of omissions. A significant part of the technical documentation comprising this vendor's TDP that is mandated by the federal VS standards was never submitted to the Documentation Review team and could not form a part of this review. No TDP documents were supplied to this review team until July 13th, very close to the end of our review.

3.2 ITA Testing Laboratory Documentation

At the outset of this review, we received qualification reports from two Independent Testing Authorities (ITAs) that Diebold had contracted with for the VS evaluations required for qualification under the 2002 federal standards.

3.3 Internal Vendor Manuals

We received a manual designed to guide installation and configuration of GEMS that was specifically restricted to Diebold technical personnel. This was named the GEMS 1.18 Server Administration Guide, Rev. 3.0 numbering 156 pages. No other Diebold documentation of pre-deployment configuration steps was provided.

3.4 Operating Manuals for Local Election Officials

We received the following Diebold operating manuals designed for voting systems customers:

GEMS

- GEMS 1.18 Election Administrator's Guide, Rev. 7.0, 536 Pages. (D0062)

- GEMS 1.18 Election Administrator's Guide, Rev. 8.0, 543 Pages. (D0087)
- GEMS 1.18 Product Overview Guide, Rev. 2.0, 13 Pages. (D0088)
- GEMS 1.18 Reference Guide, Rev. 6.0, 365 Pages. (D0063)
- GEMS 1.18 Reference Guide, Rev. 7.0, 351 Pages. (D0080)
- GEMS 1.18 Reference Guide, Rev. 8.0, 355 Pages. (D0089)
- GEMS 1.18 System Administrator's Guide, Rev 5.0, 97 Pages. (D0064)
- GEMS 1.18 System Administrator's Guide, Rev 6.0, 95 Pages. (D0090)
- GEMS 1.18 User's Guide, Rev. 12.0, 281 Pages. (D0086)
- GEMS 1.18 User's Guide. Rev 11.0, 280 Pages. (D0065)
- JResult Client 1.1 User's Guide, Rev. 1.0, 28 Pages. (D0091)

Optical Scanning – General Resources

- Ballot Specifications, Rev. 2.0, 18 Pages. (D0056)
- AccuVote-OS Hardware Guide, 124 Pages. (D0055)
- AccuVote-OS Hardware Guide, Rev 4.0, 102 Pages. (D0059)
- AccuVote-OS Hardware Guide, Rev. 5, 97 Pages. (D0066)

AV-OS Central Count

- AccuFeed 1.0 Hardware Guide, 40 Pages. (D0053)
- AccuFeed User's Guide, Rev. 1.0, 7 Pages. (D0054)
- AccuVote-OS 2.0 Central Count User's Guide, Rev. 3.0, 80 Pages. (D0058)
- AccuVote-OS Central Count 2.00 User's Guide, Rev. 4.0, 85 Pages. (D0060)
- AccuVote to Digi PortServer II (PSII), 2 Pages. (D0061)

AV-OS Precinct Count

- AccuVote-OS Pollworker's Guide, Rev. 2.0, 105 Pages. (D0067)
- AccuVote-OS Pollworker's Guide, Rev. 3.0, 116 Pages. (D0068)
- AccuVote-OS Precinct Count 1.96 User's Guide, Rev. 4.0, 209 Pages. (D0069)

TSx and Related Equipment

- AccuView Printer Module Hardware Guide, Rev. 1.0, 29 Pages. (D0077)
- AccuView Printer Module Service Guide, Rev. 1.0, 13 Pages. (D0076)
- AccuVote-TSX Hardware Guide, Rev. 8.0, 177 Pages. (D0072)
- AccuVote-TSX Pollworker's Guide, Rev. 5, 117 Pages. (D0073)
- Ballot Station 4.6 User's Guide, Rev. 1.0, 149 Pages. (D0078)
- Key Card Tool 4.6 User's Guide, Rev. 1.0, 20 Pages. (D0092)
- Upgrading Voter Card Encoder Firmware, Rev. 1.1, 4 Pages. (D0093)
- VCProgrammer 4.6 User's Guide, Rev. 1.0, 29 Pages. (D0095)
- Voter Card Encoder 1.3 User's Guide, Rev. 1.0, 30 Pages. (D0094)

CHAPTER 4

Adequacy of Support for Qualification Recommendations

4.1 ITA Qualification Testing Laboratory Reports

The federal standards under which the Diebold voting systems were evaluated accord a pivotal role to Independent Testing Authorities' (ITA) evaluations. At the time that the Diebold systems under review here received California certification (February 17, 2006), the ITA laboratory reports formed the primary basis for the "qualification" determinations made by the National Association of State Election Directors (NASED).[6] The Diebold system was reviewed and qualified by NASED under the technical standards found in the 2002 Voting System Standards (VSS)[7] and not the newer 2005 VSSG. The 2002 VSS is comprised of two volumes which specify performance standards and also the testing and examinations required for VS qualification.

At the time the Diebold systems were reviewed for federal qualification, the ITAs exercised an almost unique role and set of powers over voting systems used in this country. Given that the VS vendors have typically asserted proprietary rights and access restrictions over the source code, documentation resources, and election databases, and because State governments have generally deferred to the ITA examination regime, virtually no entities and examiners other than the ITAs have had the opportunity to evaluate the accuracy, security, and other attributes of the voting systems either for compliance with the governing standards or according to independent criteria. The ITA report is expected to reveal in extensive detail the integral parts, component operations, the scope of testing with all test results, and overall systemic functioning of the VS submitted for review. To complete these extensive evaluations, under the 2002 VSS vendors were required to submit their voting system source code, documentation, and hardware to the ITA.

---

[6] NASED no longer has the role and powers that it did at the time this Diebold system was qualified. Currently, as a result of federal regulatory changes, the U.S. Election Assistance Commission (EAC) has assumed the former powers to receive the independent laboratory reports and determine satisfaction of the federal VS criteria. This new EAC "qualification" will essentially mean national certification.

Independent testing lab supervision has been reallocated as well as. NIST now has the power to recommend accreditation for test labs to the EAC, rather than NASED. A name change has also occurred with the new certification regime: accredited test labs are now known as voting system test labs (VSTLS) rather than ITAs. Because the Diebold system received qualification under the former regulatory structure, this Report refer to the test lab reports as issuing from ITAs.

[7] The 2002 VSS have been heavily criticized for various deficiencies, including lack of technical precision and the omission of their being mandatory minimum standards for all voting systems used in any of the nation's elections. Later this year, on December 31, 2007, the 2005 Voluntary Voting System Guidelines will become the guidelines for federal certification and "[a]l previous versions of national standards will become obsolete." U.S. Election Assistance Commission, Voluntary Voting System Guidelines, http://www.eac.gov/vvsg_intro.htm (visited July 18, 2007).

As authorized, Diebold contracted with two ITAs to conduct the required VS evaluative tests: Wyle Labs and CIBER, Inc. These tests were divided into hardware and software plus documentation review. Wyle conducted the hardware and firmware reviews for the AccuVote-OS and AccuVote-TS, and CIBER reviewed the GEMS software and related software components. In addition to testing hardware, a hardware ITA tests "firmware" – the software that is embedded in voting system equipment. The software ITA normally evaluates all other software used in the system. It conducts a source code review as well as other tests. In fulfilling the federal VSS charge, the ITA completes three types of testing on the submitted voting system: functional testing, environmental testing, and the source code review.

To be a complete under the 2002 VSS, the ITA report on a component or system should include:

- ***Test Plans***: a presentation of the test plans, descriptions of the tests conducted, and the reasoning that the ITA applied to reach each of its conclusions regarding satisfaction of a particular standard or test;

- ***Matrix:*** a requirements matrix derived from the 2002 VSS; for each requirement in the matrix, the ITA notes whether the application was tested and, if so, whether the system met the requirement;

- ***Technical Documentation:*** the specified contents of the technical data package (TDP) must be submitted to the ITA; in addition to the ITA's statement that it has reviewed the TDP contents, the ITA report should demonstrate the criteria applied to determine the TDP contents' adequacy under the 2002 VSS;

- ***Longitudinal Testing Documentation:*** a review of the scope of testing and retesting that occurred over time; it is expected that an ITA report on a VS will show many instances of revision and ITA retesting; because every upgrade and version revision requires additional VS qualification, a VS normally has a substantial testing paths that can (and should be) be tracked in the report.

The SOS asked TTBR document review teams to determine whether the reports these ITAs filed are "sufficient," meaning that the reports document the tests conducted and report the results that the VS earned in the evaluations, yielding a reasonable confidence that the evaluations were conducted according to the 2002 testing standards. To this end, we examined the two labs' testing methodologies and their presentation of test data but we did not attempt to reproduce the testing, reinterpret the test data, or reconsider the ITAs' recommendation that the VS receive federal qualification. We also did not consider whether we might have conducted the testing differently.

The two ITAs pursued two quite different approaches to testing and reporting which cannot be explained simply on the basis of their different testing duties.

Table 4.0 Diebold VS Component Qualification Testing Allocation

CIBER, Inc.	***Wyle Laboratories, Inc.***
GEMS 1-18-24	AccuVote-OS Model D Precinct Count f/w 1.96.6
GEMS 1-18-24 with BallotStation 4.6.4	AccuVote-TSx with AccuView, f/w 4.6.3
Express Poll 2000/4000	AccuVote-TSx with AccuView, f/w 4.6.2
GEMS 1-18-24	VCProgrammer 4.6.1
	Key Card Tool 4.6.1
	*AccuVote-TSx with BallotStation 4.6.4

4.1.1 Wyle Report

In testing VS hardware and firmware, Wyle was required to evaluate the physical properties of manufactured components and logical correctness of computer code. The Wyle hardware testing appears to have followed standard methodologies for environmental and electrical testing of computer components and supporting electrical components. It seems, quite appropriately, to track some of the testing performed earlier by a laboratory on Diebold's behalf during product development. Wyle's report describes its methodology and other aspects of the test plans in substantial detail. It presents and analyzes the resulting test data as a basis for Wyle's recommendation to NASED that the VS be qualified under the 2002 VSS.

The Wyle report[8] adequately tracked the version numbers of the components under review. In evaluating the source code for the firmware under review, Wyle seems to have relied extensively on automated tools for finding deviations from coding standards or uses of prohibited flow-of-control constructs. Perhaps for this reason, the results of its source code examination seem to emphasize form.[9]

It appears that Wyle properly noted and tracked the version designations of specific Diebold election products as it tested and reported on the performance of Diebold products.

Wyle did note that it had not tested certain aspects of the AccuVote-TSx and AccuVote-OS. Its ITA report indicates the VS was not tested, on the following points:

- 2.2.5.2.3 Test for the capability for a jurisdiction to designate critical status messages
- 2.2.8.2(m) Support of cumulative voting.
- 2.2.8.2(n) Support of ranked over voting.
- 2.2.10 Telecommunications (Hardware Functional & Software System Level Test.)
- 3.2.8.2 Generation of output reports at the device, polling place and summary levels, with provisions of administrative and judicial subdivisions as requirement (sic) by the jurisdiction.
- 3.4.5(c) DRE and paper-based precinct count systems and supporting software respond to operational commands and accomplish the functions of consolidation of vote selection data from multiple precinct-based systems, generate jurisdiction-wide vote counts, store and report the consolidated vote data.
- 3.4.5(c) DRE and paper-based central count systems and supporting software respond to operational commands and accomplish the functions of consolidation of vote selection data from multiple counting devices generate jurisdiction-wide vote counts, store and report the consolidated vote data.
- 4.3(a) and (b) During an election, the integrity of vote and audit data is maintained and protected against any attempt at improper data entry or retrieval.

On the basis of the materials we received it appears the Wyle Lab conducted a broad range of tests and engaged in detailed analytic reporting on the adequacy of the test results. In our view, taken as a whole the Wyle reports provide substantial evidence in support of its recommendation that the Diebold VS receive federal qualification under the 2002 VSS.

---

[8] *Hardware Qualification Testing of the Diebold Election Systems AccuVote Optical Scan Model D Precinct Ballot Counter (Firmware Release 1.96.6).*

[9] While some coding standards aid readability, maintainability and security, and may reduce the likelihood of several classes of bugs, automated code analysis has some notable deficiencies and cannot suffice for comprehensive code assessment.

4.1.2 CIBER Report

In contrast to the test reports from Wyle, the CIBER Qualification Test Report (D0044) issued for GEMS 1.18.24 is brief and perfunctory. It fails to include basic information that is essential to establishing the functionality, reliability, security and certification status of GEMS itself, or of proposed or anticipated GEMS configurations. The report's 16 pages are devoted primarily to broad generalizations drawn from the 2002 VSS and do not sufficiently elaborate on any specifics of the testing methodology, test results or bases for its recommendation that GEMS 1.18.24 be qualified.

The report's cursory recitals fail to offer any basis for assessing the adequacy and repeatability of any testing or analysis that CIBER may have performed on the software. At no point does it demonstrate that an adequate evaluative review of GEMS and its associated systems was undertaken. The CIBER report does not describe the VSS-required vendor submission of various test plans, nor does it discuss conducting detailed tests or the test results that were thereby obtained. In scope, substance, and style, the CIBER report on the GEMS software resembles Wyle's Executive Summary of its qualification report for the AccuVote-TSx with AccuView Printer Module.[10]

While the length of CIBER's report alone is not dispositive as to its adequacy, it is interesting to note that a subsequent report issued by CIBER and documenting a highly circumscribed test effort (involving some report scripting code resident in a Diebold voting device), spans 13 pages—which is nearly as long as the entire 16-page GEMS qualification test report. (D0042). Because of their extreme brevity and lack of particulars, the CIBER reports do not permit the documentation review team to discern what kind of testing was completed. The reports simply assert the adequacy of the system and its component parts under review rather than establish a basis for understanding any testing and analysis that was completed.

4.1.3 The Technical Data Package: Selected Analysis

In assessing the vendor documentation and the ITA testing, we were able to review part of the vendor-supplied Technical Data Package (TDP) that the 2002 VSS mandates for submission when the ITA qualification review is initiated. We were accorded access to the TDPs for GEMS and for the AccuVote-TSx. The vendor did not submit to the TTBR specialized TDP documentation for the optical scanning systems or for any software other than GEMS.

While the submissions were not structured according to the organization of the TDP definition found in the VSS, each TDP included a "TDP trace" document which was helpful in mapping the documentation provided in the TDP to the subject-specific sections of the TDP format.

Many TDP sections refer to the customer documentation as a source of technical or operational information. In addition to missing TDP materials for several Diebold systems, some of the customer documentation that is incorporated by reference into the technical documents that we reviewed was not submitted for the TTBR. The gaps in vendor documentation relate to many aspects of the Diebold voting systems under review.

Accuvote TSx The TDP for the AccuVote-TSx included extensive information about the design, manufacture, assembly, and testing of these DRE voting machines. The specifications as provided by the component vendors include those for many electrical and electronic components that are integrated into

[10] Compare, *e.g.,* D0100 *Wyle Test Report: Hardware Qualification Testing of the Diebold Election Systems AccuVote-TSx DRE Voting Machine with AccuView Printer Module, p. 5,* Section 1.3 "Summary." The Qualification Test Report (D0044).

AccuVote-TSx. These parts from other vendors are subject to quality and conformance standards for manufactured parts; technical information is supplied documenting the internal controls for hardware development and manufacture. These TDP documents also include descriptions of Diebold's programs for establishing and maintaining relationships with its suppliers and seeking to ensure a reliable stream of quality components.

The TSx technical documentation supplied in the TDP covers the spectrum from the very detailed and concrete, for example. the placement of screw holes for mounting the motherboard of the TSx, to statements that are abstract and aspirational, such as a corporate vision statement comprised of a series of sentences starting with "*We Believe....*" As such, the TDPs provided by Diebold are not so much a set of documents created or adapted to address the VSS requirements for a TDP as they are a large, over-inclusive set of pre-existing documents submitted to cover all the subject areas prescribed for a TDP.

Some details of the Diebold TDP submission bear a further note:

- The portion of the GEMS TDP devoted to what Diebold calls "operating system-level safeguards intended to protect against tampering"[11] is simply an unattributed reproduction of a chapter of the 1995 Microsoft publication entitled *MS Windows NT Workstation 4.0 Resource Guide.*[12] The chapter is called "Windows NT Security,"[13] and discusses the overall security framework of the Windows NT operating system. While informative, the Microsoft guide contains a broad overview of system and network security models, as implemented in Windows NT. It does not indicate which, if any, of the Windows NT security features described in the chapter are employed in securing a GEMS server. The information in the chapter is readily available, public information that has little direct applicability to the election management context. It is the sort of information that a qualified testing organization can be expected to know or be able to locate if needed. Further, any ITA seeking to do a realistic assessment of GEMS would base that assessment on the security features of the actual platform on which the system is being tested (Windows 2000 Server) rather than those of a predecessor platform (Windows NT 4.0).[14]
- The exclusive discussion in the TDP of the platform security specifications of Windows NT 4.0 seems out of place with the repeated assertions in the TDP that GEMS was designed for Windows XP, the fact that the Server Administrator's Guide and other documentation describes installing GEMS on Windows 2000 Server, and the fact that the GEMS server provided to the TTBR study was running Windows 2000 Server. For a discussion of platform configuration issues, see Section 4.3 *Configuration*.
- One troubling finding based on a comparison of the TDP and the customer documentation supplied by Diebold is that the security policy presented in the Diebold customer documentation differed significantly with the mandatory client security policy that Diebold submitted to the ITA. See Section 6.1 for an extensive discussion of this issue.

---

11 *GEMS 1.18 Technical Data Package – System Functionality Description*, p. 2-4.

12 The original Microsoft publication is available at: http://www.microsoft.com/technet/archive/ntwrkstn/reskit/default.mspx, accessed 7/17/2007.

13 The chapter included in the GEMS TDP is available at: http://www.microsoft.com/technet/archive/ntwrkstn/reskit/security.mspx, accessed 7/17/2007.

14 Note that the Microsoft publication from which the appendix is copied contains the following notice: "© 1995 Microsoft Corporation. All rights reserved. No part of this document may be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose, without the express written permission of Microsoft Corporation." No such written permission is included in the GEMS TDP.

4.1.4 Are the ITA Recommendations to NASED Adequately Supported By the Documentation?

The CIBER report does not support a confident conclusion that the testing laboratory evaluated the source code or the technical documentation for its compliance with federal 2002 VSS. The Wyle hardware and firmware report is substantially more extensive than the CIBER report but limitations in the documentation supplied for our review prevent us from fully assessing the adequacy of the Wyle evaluations for supporting NASED qualification.

4.2 State Certification

The central regulatory charge to the Secretary of State incorporates interpretive breadth and the possibility for varying standards over time. The primary statutory charge to the Secretary of State is found in California Election Code § 19205 (emphasis added):

> The Secretary of State shall establish the specifications for and the regulations governing voting machines, voting devices, vote tabulating devices, and any software used for each, including the programs and procedures for vote tabulating and testing. The criteria for establishing the specifications and regulations shall include, but not be limited to, the following:
>
> **(a)** The machine or device and its software shall be *suitable for the purpose* for which it is intended.
>
> **(b)** The system shall *preserve the secrecy of the ballot*.
>
> **(c)** The system shall be *safe from fraud or manipulation*.

Diebold's voting system has traversed a somewhat rocky California regulatory road which has included certification, decertification, and re-certification. The Certificates that have issued permitting the VS to be used within the State have carried some significant restrictive conditions. The State certification system records provided for this review do not reveal any independent source code testing or repetition of the VSS review at the time the Diebold VS was presented for re-certification, but the records are not extensive or illuminating and a source code or other extensive review might have occurred.

4.2.1 California State Certification Process

The certification testing of Diebold systems conducted by the State of California and its designees was successful in validating crucial aspects of election system performance and uncovering problems that needed to be addressed prior to certification by the Secretary of State. Though the scope of the testing was more circumscribed and the access to some types of vendor information more limited than was the case for ITAs testing the same systems, California's testing was arguably more thorough and helpful in advancing the objectives of accuracy, reliability and security than the testing performed by ITAs.

The reports[15] issued by those entities testing the Diebold systems on behalf of the State were far superior to those issued by CIBER, the ITA hired by Diebold to perform qualification testing of its GEMS product. In contrast to the CIBER reports covering the same systems, the State reports have thorough system configuration information[16] and discussion of test objectives, methodologies, and limitations. Also unlike the CIBER reports, the State reports included detailed information about the nature and frequency of failures and anomalies, and included some primary test data that gave an accessible illustration of the testing procedures.

Perhaps as important as the State testing itself was the apparent success with which the State was able to work with Diebold to make beneficial changes to its products in response to unsatisfactory test performance. This cooperative approach to resolving problems with election technology may help bring about additional changes to Diebold systems to better advance the accuracy, reliability and security of

---

[15] See *Staff Review and Analysis Report*, Prepared by the Secretary of State Elections Division, November, 14, 2005, and Steven V. Freeman, *Certification Test for the Diebold Election Systems, Inc. (DESI) GEMS 1.18.24, AV-OS 1.96.6, AV-TSX 4.6.4 with AccuView Printer Module, and Voter Access Card utilities*, November 11, 2005.

[16] See Steven V. Freeman, *Certification Test for the Diebold Election Systems, Inc. (DESI) GEMS 1.18.24, AV-OS 1.96.6, AV-TSX 4.6.4 with AccuView Printer Module, and Voter Access Card utilities, Attachment A*, November 11, 2005.

elections in California.

Some room for improvement exists in the formal presentation and transmittal of procedural safeguards intended to address issues uncovered in testing. Future reports might better serve the Secretary of State and election officials by consolidating express recommendations for usage procedures intended to mitigate problems discovered in testing. The reports we examined had mixed together proposed policy statements with the test results and technical findings.

4.3 Configuration Issues: Nonconformity with Certified Configuration

In the VS testing and certification regime, the most extensive testing is reserved for voting system components that are designed by the election system vendor to handle election-related functions. In the case of software components, the testing involves a review of the source code. Some "commercial off the shelf" (COTS) software that is distributed with the VS is subject to less rigorous testing standards, though it must be tested as part of overall system testing.

In evaluating and comparing the vendor documentation, ITA test reports, certification documents and the vendor-configured Diebold VS submitted for the TTBR, we found configuration differences that raise questions about the consistency with which certain voting system components are provided to customers in the configurations for which they were certified. By analyzing the results of a configuration audit we conducted on the Diebold VS submitted for the TTBR, we discovered some significant inconsistencies between parts of the VS that are or were:

- described in the vendor-supplied documentation;
- the subject of a successful qualification or certification review;
- actually being deployed in California counties; and those
- submitted by the vendor for TTBR testing in California.

The table comparing the range of differences follows this section and is identified as ***Table 4.3, Configuration Comparison.***

Some discrepancies are of comparatively little concern. Certain COTS support applications on the GEMS server, for example, have been deployed with minor version variations. While not ideal, these variations do not necessarily have the same significance as other discrepancies discussed below.

Because of the differences in configuration we present below, we are unable to conclude that the vendor submitted to the ITA the requisite documentation germane to the sought VS certification for a particular configuration. Nor can we conclude that the officially qualified and certified VS (whether for federal or State reviews) is the exact VS that is currently being deployed in California counties that use Diebold equipment. Determining which California counties, if any, are currently using the Diebold VS in an uncertified configuration is beyond the scope of this review.

4.3.1 Uncertified Components and Configurations

As discussed below, we find that certain Diebold VS configurations that are widely used in California counties may not have been federally qualified or State certified. For the affected counties, integration of uncertified VS components among certified components may render the voting system as a whole uncertified for use in California.

4.3.1.1 JResult Client

As part of its GEMS product, Diebold's distributes a Java-based application program called JResult Client. JResult Client is used for periodically generating visually-appealing updated reports of election results for public display and for posting on the Web as tabulation proceeds on election night.[17] JResult Client is pre-installed on GEMS servers prior to delivery to customers.[18] It can also be installed and used on Windows-based PCs other than the GEMS server.[19] We believe that the presence of JResult Client on GEMS servers in California raises a number of very serious concerns about the integrity of the GEMS servers deployed in California, as well as their legal status.

As illustrated below, there is no evidence in any documentation we have reviewed that JResult Client has been submitted, with the required documentation and source code, for testing by an ITA. There is no evidence that it has qualified under federal standards, been submitted for certification in California or been certified for use in California. The California certification of GEMS 1.18.24 was granted with the condition that "[n]o additional software developed by the Vendor other than that specifically listed in this certificate shall be installed on a computer running GEMS version 1.18.24."[20] These facts raise substantial doubt as to the certification status of any GEMS server in use in California that includes JResult Client. As such, it is possible that GEMS servers in use throughout California are used in an uncertified configuration.

JResult Client is installed on and can be run on the GEMS server machine, a separate Windows workstation, or both.[21] As its documentation, configuration and persistent interaction with GEMS make clear, JResult Client is a GEMS component and not a COTS item. JResult Client is a Diebold-developed application, written and deployed to facilitate election management activities on the GEMS server. The documentation specifies that a Diebold technician or an election administrator shall install JResult Client as a "component" of the GEMS product via the GEMS install wizard.[22] The JResult Client User's Guide is distributed with the other GEMS documentation.[23] JResult Client cannot function apart from its interaction with the GEMS Results Server. JResult Client is also included as a Diebold product in the Diebold Election Systems' Bugzilla database.[24]

JResult Client does not appear on the list of software submitted for ITA testing[25] or recommended for federal qualification.[26] It does not have a Qualification Number. It was not submitted for California certification, does not appear in the detailed configuration specification included in the California certification testing report,[27] and does not appear on the California certificate.[28] It is installed as part of

---

17 *GEMS 1.18 System Administrator's Guide, Revision 5.0,* p. 1-2.

18 *GEMS 1.18 Server Administrator's Guide, Revision 3.0,* p. 2-51.

19 *GEMS 1.18 Server Administrator's Guide, Revision 3.0,* p. 1-3.

20 Secretary Of State of California, *APPROVAL OF USE OF DIEBOLD ELECTION SYSTEMS, INC. DRE & OPTICAL SCAN VOTING SYSTEM,* Section 4(a), February 17, 2006.

21 "JResult Client may be installed and operated on the GEMS server, in addition to the designated JResult Client machines." *GEMS 1.18 System Administrator's Guide, Revision 5.0,* p. 2-7.

22 *GEMS 1.18 Server Administrator's Guide, Revision 3.0,* p. 2-51.

23 *GEMS 1.18 Product Overview Guide, Revision 2.0*, p. 2-2.

24 *Bugzilla User's Guide, Revision 1.0,* p 9. (This document is a customized guide for use within Diebold Election Systems, Inc.)

25 *Software Functional Test Report, Diebold Election Systems GEMS 1-18-24, Version 1.0*, CIBER, Inc., August 3, 2005, pp. 2-3.

26 *Software Functional Test Report, Diebold Election Systems GEMS 1-18-24, Version 1.0,* CIBER, Inc., August 3, 2005, p. 8.

27 *Staff Review and Analysis Report*, Prepared by the Secretary of State Elections Division, November, 14, 2005,

GEMS and was present, in compiled form, on the GEMS server provided by Diebold to CIBER[29] and the GEMS server provided by Diebold for the TTBR study.[30] This server, Diebold representatives assured us, was configured in the same manner as the other GEMS servers sold to California election jurisdictions.

A directory listing included in *Addendum 5* to the CIBER testing report for GEMS 1.18.22 indicates that the compiled JResult Client application files were present on the GEMS server submitted for ITA testing. This is the only reference to JResult Client contained in the CIBER report. It is not clear why CIBER did not flag this software in its report or seek more information about JResult Client from Diebold prior to recommending qualification of the system. Neither the source code nor the system design for JResult Client appear to have been reviewed by CIBER and were likely not submitted by Diebold for CIBER's review. The CIBER report lists programming languages employed in the source code they reviewed, and Java appears nowhere on this list. The GEMS TDP includes no description of the JResult Client design or testing and includes no coding standards for Java programming.

While the non-certification of JResult Client in California, by itself, compromises the integrity of elections in California jurisdictions using GEMS, we discovered several specific issues with JResult Client that we believe make its presence and use on election equipment in California cause for very serious concern.

The first issue is that the GEMS documentation describes an approved use case in which JResult Client, running on the GEMS server, posts election results to an FTP server connected to the Internet.[31] The establishment of such a link, however indirect, between the GEMS server and the Internet should be thoroughly examined as a matter of election security.

The second issue of great concern is that JResult Client and the GEMS Results Server interact with a common resource on the GEMS machine, or on a networked machine.[32] An examination should be done into potential concurrency problems that could cause the processes to collide and threaten the stability of the GEMS server during election tabulation.

Also of concern is the specific JResult Client configuration as installed on GEMS servers. Both the file listing in the CIBER report addendum mentioned above, and our own configuration audit of the GEMS server provided to the TTBR reveal that the GEMS installation script copies both individual Java class files (bytecode files) and a JAR file containing the class files to the JResult Client installation directory on the GEMS server. This is not only a configuration management problem, but a potential vector for an attack. An attacker with access to the GEMS server could, for example, copy modified class files to the GEMS server in place of the original files. These potentially malicious class files could be executed in place of the code in the JAR file by making a small change to the CLASSPATH environment variable or to a batch file that invokes the JResult Client program. This malicious code may evade detection because anyone verifying the signature on the JAR file would find that it has not been modified and conclude that the JResult Client application in use on the GEMS server has not been modified. The fact that this configuration defect was not noticed or not reported by CIBER is also cause for concern about the thoroughness of their testing of GEMS.

---

and Steven V. Freeman, *Certification Test for the Diebold Election Systems, Inc. (DESI) GEMS 1.18.24, AV-OS 1.96.6, AV-TSX 4.6.4 with AccuView Printer Module, and Voter Access Card utilities*, November 11, 2005.

28 Secretary Of State of California, *APPROVAL OF USE OF DIEBOLD ELECTION SYSTEMS, INC. DRE & OPTICAL SCAN VOTING SYSTEM*, Section 1, February 17, 2006.

29 *Addendum 5* of the to the CIBER testing report for GEMS 1.18.22.

30 *See* Section 4.3.2 *Configuration Audit, infra*.

31 *GEMS 1.18 Election Administrator's Guide, Revision 8.0*, p. 16-15.

32 *GEMS 1.18 Election Administrator's Guide, Revision 8.0*, p. 4-128.

Finally, as best can be determined from the materials provided, the JResult Client program (and its accompanying documentation) were not present on the GEMS server that Diebold submitted for certification testing in California. Given the presence of JResult Client on the GEMS server this team analyzed, and the internal GEMS configuration manual's instructions to Diebold technicians to install JResult Client as part of the standard GEMS installation, the question arises of why the vendor excluded JResult Client from its California certification submission.

4.3.1.2. COTS, Windows OS, and Other Configuration Irregularities

In assessing the VS configurations, we found that Diebold systems were deployed with a range of different COTS software components other than those that were tested and those that appear in Diebold's internal and customer documentation. There were several, somewhat minor applications that were slightly different from the tested and documented versions (file archive utility, audio recorder and codec).

Some COTS applications were present on the TTBR GEMS server that were not part of any tested and documented configuration. These programs should not be installed on any election server at all (e.g., Outlook Express, NetMeeting).

Windows OS Version Differences More worrisome than the variations in COTS application programs, however, is the fact that the GEMS documentation, testing, certification and deployment documents do not agree on which Windows operating system versions GEMS should or could be installed. As explained above, the TDP section on platform security exclusively discusses Windows NT 4.0.. Elsewhere in the TDP, it notes that GEMS was designed to be run on Windows XP. Yet the GEMS internal and customer documentation describes how to install, configure and maintain GEMS on the Windows 2000 Server and no other system. Finally, the ITA test report indicates that their testing was done using GEMS on Windows 2000, and the GEMS server supplied to the TTB ran Windows 2000.

Unlike small version changes in the minor COTS utility applications described above, differences in an operating platform generate significant implications for reliability and safety that need to be explored and thoroughly tested. Though the Windows operating systems listed are related to one another and share many features and capabilities, GEMS needs to be designed, tested, documented and deployed for a common operating system or systems. Crucial, ongoing configuration and updating tasks essential to securing the GEMS server, for example, are performed differently for different Windows versions, and each one needs to be tested and documented if it is to serve as a platform for the GEMS application. (Though we have no way of knowing the cause of the severe mis-configuration of crucial security settings on deployed GEMS Windows 2000 machines that we outline below, such mis-configuration could be the result of a Diebold installation procedure that does not properly follow the specific steps required to configure certain settings on Windows 2000 machines.)

4.3.2 The Configuration Audit

We performed a configuration audit of the GEMS server supplied for the TTB review (the "TTBR GEMS server"). Diebold technical representatives confirmed that it was assembled by Diebold and that they configured the hardware and software exactly as it would be upon delivery to a California county that was purchasing such equipment. We then compared the configuration of the TTBR GEMS server to the configuration set forth in the *GEMS 1.18 Server Administrator's Guide, Revision 3.0*, and to the configuration of the GEMS server evaluated by CIBER for the August 3, 2005 report recommending GEMS 1.18.24 for NASED qualification.

When compared, the configurations showed many similarities, but some variations were noticeable. At least a few variations may figure in the relative accuracy, reliability and security of the Diebold election system. Additionally, some of these variances appear to deviate from the certified configurations for use in California elections.

The terms of the VSS reflect the accepted fact that components of computer systems cannot be evaluated without reference to the other components with which they must interact. Any faulty or malicious piece of software or hardware in a computer system could compromise the entire system. Furthermore, two separate components in a system may be incompatible, may interfere with one another or may not work well together. Configuration changes may bring to life latent bugs in a system component that were not evident before. In short, configuration management is crucial, and is accorded an entire chapter of the VSS.[33]

The *GEMS Server Administrator's Guide* is a document available to and used by Diebold technical personnel to configure systems prior to releasing them to the election system customers. It walks through step by step procedures for installation of the operating system and software, and covers setting various settings in the operating system, the GEMS application, and other aspects of the software environment. The document clearly states that it is not permitted to circulated beyond vendor personnel, and thus is unavailable to customers. Furthermore, the topics and configuration choices covered in the *Guide* are not explained to the election official customers in the customer documentation. For this reason, correct configuration of the GEMS server by the Diebold staff prior to delivery of the machine is essential; the customer will not have the needed guidance to remedy any mistakes, and may not even detect them..

We found several important areas in which the GEMS configuration settings in the *GEMS Server Administrator's Guide* were not adhered to by Diebold personnel when they set up the TTBR GEMS server. This resulted in significant configuration deviations from the documented specifications. We also found several areas in which the *GEMS Server Administrator's Guide*, even if followed exactly, does not yield an acceptable configuration for use by an election jurisdiction.

Several software packages that were to be installed according to the *GEMS Server Administrator's Guide* were not installed on the TTBR GEMS server. Additionally, as the configuration audit table illustrates, several applications were present on the TTBR GEMS server that were not called for in the *GEMS Server Administrator's Guide* and which are not appropriate for a secure election server.

Perhaps most importantly, we found crucial security settings that were not configured correctly on the TTBR GEMS server. The security policy on the TTBR GEMS server was not set up to log any security events through the Windows event logs. There were no restrictions configured to limit the access by ordinary GEMS users to core operating system functions and settings. No password policies or other security policies were implemented to bring the security of the system up to an acceptable level of confidence in mission critical, sensitive election equipment.

The TTBR GEMS server, as delivered and configured, was not suitable for use by any county without significant reconfiguration. Because these crucial aspects of the configuration of the GEMS server are to be implemented by Diebold personnel and are not described in any customer-accessible manual, an election jurisdiction would have no source of information on how to bring about the proper, certified configuration, and may indeed not be able to recognize that the configuration is not correct.

---

[33] VSS, Vol. I, section 8; configuration audits are delineated in section 8.7.

Table 4.3 Configuration Comparison

The following table presents the types and versions of various components of the Diebold VS encountered in different documents and deployed systems examined for the TTBR.. The table compares the equipment submitted by Diebold to the TTBR, the systems described in the GEMS Server Administrator's Guide, the systems reviewed during ITA testing, the systems reviewed during certification testing in California, and the systems as ultimately certified for use in California. Individual names and version designations are given with as much specificity are possible, and thus may not be uniform for all entries in a row. See section 6 of this report for information on security-related configuration issues.

	TTBR Equipment Configuration Audit	GEMS Server Administrator's Guide	ITA Reviewed Configuration	NASED Qualified as N-1-06-22-22-002	Cal. Test Configuration Report of 11/11/2005	SoS Certified on 2/17/2006
GEMS Software	1-18-24	Written to encompass several GEMS versions.	1-18-24	1-18-24	1-18-24	1-18-24
AVOS Precinct F/W	1.96.6	N/A	1.96.6 (D0044)	1.96.6	1.96.6	1.96.6
AVOS Central F/W	2.0.12	N/A	2.0.12 (D0044)	2.0.12	2.0.12	2.0.12
AVTSx with AVPM	Bootloader BLR7-1.2.1 Windows CE WCER7-410.2.1 Firmware 4.6.4	N/A	Bootloader BLR7-1.2.1 Windows CE WCER7-410.2.1 Firmware 4.6.4	Bootloader: Not specified. Windows CE: Not specified. Firmware 4.6.4	Bootloader: Not specified. Windows CE: Not specified. Firmware 4.6.4	Bootloader: Not specified. Windows CE: Not specified. Firmware 4.6.4
Key Card Tool	4.6	N/A	4.6.1 (D0044)	4.6.1	4.6.1	4.6.1
VCProgrammer	4.6	N/A	4.6.1 (D0044)	4.6.1	4.6.1	4.6.1
Voter Card Encoder	Spyrus Vote Card Encoder: 1.3.2	N/A	1.3.2 (D0044)	1.3.2	Not Specified.	1.3.2
JResult Client	1.1.3	1.1.3	None (D0044)	None.	None.	None.
GEMS readme.html	Not Present	Present	Not Specified.	Not Specified.	Not Specified.	Not Specified.
Compiled Help	Incomplete/unusable	Present	Not Specified.	Not Specified.	Not Specified.	Not Specified.

Note: The Accuvote-TSx certified in California uses BallotStation 4.6.4, which was approved by Ciber (D0045) for use with GEMS 1.18.24. During the period of the testing, BallotStation 4.6.4 was designated version 4.6.3.12. The version number was promoted to 4.6.4 upon satisfactory completion of the ITA testing.

GEMS Operating Platform

	TTBR GEMS Installation	GEMS Server Administrator's Guide	ITA Reviewed Configuration	NASED N-1-06-22-22-002	Cal. Test Configuration	SoS Certified 2/17/06
Operating System	Microsoft Windows 2000 Server, v. 5.0.2195 SP4 Build 2195	Microsoft Windows 2000 Server SP4	Microsoft Windows 2000 Server, v. 5.0, SP4	Not Specified.	Microsoft Windows 2000 Server, v. 5.0, SP4 "with additional patches for SP5"	Not Specified.
H/W Platform	Dell PowerEdge 2600	Dell PowerEdge 2500 PIII @ 1.13GHz (the "medium" platform)	X86 Family 130,616 KB Ram (*sic*) (D0044)	Not Specified.	Dell PowerEdge 600SC Pentium 4 @ 1.8 GHz	Not Specified.
BIOS	Phoenix ROM BIOS PLUS Version 1.10 A09	Not Specified.	Not Specified. (D0044)	Not Specified.	Not Specified.	Not Specified.
RAM	1 GB	1 GB	128 MB (D0044)	Not Specified.	1 GB	Not Specified.
Graphics Card	ATI RAGE XL PCI	Video card capable of 1024x768	Not Specified. (D0044)	Not Specified.	Not Specified.	Not Specified.
NIC (#1)	Intel PRO/100 S Server Adapter Ethernet 802.3	Intel PRO/100 S	Not Specified. (D0044)	Not Specified.	Not Specified.	Not Specified.
NIC (#2)	Intel PRO/1000 XT Ethernet 802.3	None.	Not Specified. (D0044)	Not Specified.	Not Specified.	Not Specified.
3½ " Floppy Drive	OEM	OEM	Not Specified. (D0044)	Not Specified.	OEM	Not Specified.
Optical Media Drive	H-L Data Storage Model GCC-4240N CD-RW/ DVD-R	24X IDE CD-ROM	Not Specified. (D0044)	Not Specified.	PLEXTOR CD-R PX-W1210S (SCSI)	Not Specified.
Tape Drive	Dell STD2401LW	Internal Tape Backup Unit, 20/40 GB	Not Specified. (D0044)	Not Specified.	Not Specified.	Not Specified.
Local Fixed Disks	4 x 100GB SCSI (NTFS)	3 x 18 GB 10 KRPM Ultra 160 SCSI Hard disk	Not Specified. (D0044)	Not Specified.	20 GB IDE	Not Specified.
SCSI Card	Adaptec 39160/3960D – Ultra160 SCSI	Not Specified.	Not Specified. (D0044)	Not Specified.	Not Specified.	Not Specified.
RAID Controller	DELL PERC 4/Di	4. PERC3, DC, 128MB hard drive controller, 1 internal and 1 internal channel	Not Specified. (D0044)	Not Specified.	Not Specified.	Not Specified.
Sound	Creative	Not Specified.	Not Specified.	Not Specified.	Not Specified.	Not Specified.

Card	SoundBlaster Audigy 2ZS		(D0044)			
IEEE 1394 Controller	Creative SoundBlaster (Driver v 1.82.0, Firmware 0.0.0)	Not Specified.	Not Specified. (D0044)	Not Specified.	Not Specified.	Not Specified.
Printer	HP LaserJet 1022n	HP LaserJet 9000	Not Specified. (D0044)	Not Specified.	HP LaserJet 1020	Not Specified.
	COTS Software					
	TTBR Equipment Configuration Audit	**GEMS Server Administrator's Guide**	**ITA Reviewed Configuration**	**NASED Qualified as N-1-06-22-22-002**	**Cal. Test Configuration Report of 11/11/2005**	**SoS Certified on 2/17/2006**
MDAC	2.8.0.1022.3	2.0	"mdac_typ.exe" (only filename is given.)	Not Specified.	Not Specified	Not Specified.
Java Virtual Machine	MS JVM 5.0	MS JVM 5.0	"msjavax86.exe" (only filename is given.)	Not Specified.	Not Specified.	Not Specified.
JET DB Engine	4.0.9025.0	4.0	Not Specified. (D0044)	Not Specified.	Not Specified.	Not Specified.
File Compression Utility	WinZip 8.1 SR-1 (5266)	WinZip 8.0	Not Specified. (D0044)	Not Specified.	WinZip 8.1 SR-1	Not Specified.
Audio Recorder/ Editor	None.	Adobe Audition v 1.0	Not Specified. (D0044)	Not Specified.	Adobe Audition v 1.0	Not Specified.
VOIP/Videoconferencing client software	Microsoft NetMeeting 3.0	None.	Not Specified. (D0044)	Not Specified.	None.	Not Specified.
Email Client	Outlook Express 6.00.2800.1106	None.	Not Specified. (D0044)	Not Specified.	None.	Not Specified.
Media Player	Windows Media Player 6.4.09.1129	None.	Not Specified. (D0044)	Not Specified.	None.	Not Specified.
mp3 CODEC	Fraunhofer IIS MPEG Layer-3 codec	Lame ACM mp3 CODEC V 0.8.0 – 3.92	Not Specified. (D0044)	Not Specified.	Not Specified.	Not Specified.
PDF Creation Software	Adobe Acrobat Standard v 6.0	Adobe Acrobat Standard v 6.0	Not Specified. (D0044)	Not Specified.	Adobe Acrobat 6.0.0.2003051900	Not Specified.
Optical Media Writing Software	Nero 6 Ultra Edition	Nero 6 Ultra Edition	Not Specified. (D0044)	Not Specified.	Nero ROM Burning Suite, Version 6	Not Specified.
Asian Fonts	None.	Should be installed by Diebold tech prior to delivery	Not Specified. (D0044)	Not Specified.	Not Specified.	Not Specified.

CHAPTER 5

Sufficiency of Documentation

The California Secretary of State requested the Documentation Review teams to determine whether the vendor documentation is ***sufficient.*** Here, we considered whether the documentation adequately supports a county Registrar of Voters' deployment of the electronic voting system (VS) in a manner that empowers the election officials to administer elections accurately, reliably, and with appropriate security and ballot privacy throughout the election processes. Adequate documentation will assist officials in avoiding errors that can cause systemic problems and alert them to strategies they can employ to prevent and detect malicious tampering with the election equipment or vote totals.

Documentation sufficiency for county election purposes turns in part on whether the materials are ***usable*** by the intended audiences. The three key audiences to consider are county election officials, poll workers, and voters.

5.1 Usability Analysis in Voting Systems

Usability analysis is known within computer science and other technical fields as assessments that evaluate whether the materials – hardware, software and documentation – allow the intended human user to perform the expected or desired operations efficiently and confidently. For instance, an automobile that required the ignition key to be inserted on the left side of the steering wheel instead of the right would present American drivers with difficulties in locating the ignition and in inserting the key (using the unexpected left hand). These unexpected departures might also unsettle the driver's comfort level in the car, reducing concentration and pleasure in driving. The auto design world thus takes account of the "human factors" – the ultimate users' expectations, their physical and psychological assets and limitations, as well as the demands of the particular contexts where the automobile "product" will be used.

When designers of a product or its documentation are sensitive to usability considerations, the contexts of its deployment are evaluated along with the actual users' wants, needs, and limitations. Documentation that meets high standards of usability, then, will be tailored to a particular user audience whose needs and wants will be well-researched and well-served For instance, documentary materials discussing the DRE touchscreen voting machine (Diebold's TSx) will not be particularly effective if some hypothetical universal audience is the focus. Rather, county election administrators and technicians should receive different information and possibly a differently organized presentation than that for poll workers or voters. Each target audience has distinct needs, owing to its different duties with regard to elections.

Usability analysis in elections has thus far tended to focus on the voter. For instance, a Brennan Center report reviews studies documenting the statistical differences between different types of voting

systems with regard to the consequences for voters' ability to complete a valid ballot.[34]

In the analysis that follows, we focus predominantly on the usability of vendor-supplied electronic format documentation with regard to the two target audiences who play critical roles in election administration: county election officials and poll workers. These audiences have received little attention in the nascent election usability field, with far more concern being placed on voters' needs. Yet, as Quesenbery notes, voters' needs cannot be well-served if election officials and technicians are not well-served in preparing the election. [35]

5.2 Usability of Documentation by Election Officials

The conscientious county election manager, like managers elsewhere, desires to demonstrate a quality job performance in assigned work duties. Given the non-negotiable election calendar, efficiency and timeliness of performance matters can matter greatly. Greater media and public scrutiny of election operations may lead to significant job stress, because of a perceived need for election administrative operations to move very smoothly. If unexpected technical events or administrative problems with, for instance, ballots occur, a cascade of problems can ensue simply because the calendar dates for other tasks are dependent upon effective completion of predicate tasks. The election administrative process, then, especially in urban counties, requires a series of intersecting functions to meet precisely on time with little more flexibility than that of trapeze artists. The consequences of failed elections can not only mean job sanctions including termination, but the potentiality of individual civil or criminal penalties. Thus, stakes are very high.

In order to manage an election that produces prompt reporting of accurate, verifiable results on relatively new electronic voting equipment, officials will be often find themselves facing an intense learning curve. Questions and concerns will arise frequently. Given the election calendar fast, accurate responses are needed to the questions about the VS technology, ranging from supplies and maintenance to intensive Election Day operations.

This understanding of election officials' managerial situations leads to several criteria for evaluating the adequacy of VS documentation:

- ***Speed of use***: Can answers to questions be found quickly with little wasted time scanning irrelevant material?
- ***Accuracy and consistency***: Does the documentation provide one answer that is correct, or several conflicting answers?
- ***Clarity***: Is the documentation written with unambiguous, relatively easy to understand language?
- ***Risks***: Where a given procedure holds great risks for achieving a successful election if not performed precisely right, does the documentation flag it with visual cues and exceptionally clear writing?

---

[34] The Brennan Center of Justice report, T*he Machinery of Democracy: Usability*, reviews a the literature on voter usability. *See* *www.brennancenter.org/dynamic/subpages/download_file_36340.pdf*.

35 *See* Whitney Quesenbery, *Position Paper for UPA Voting and Usability Workshop 2004, found at* *www.upassoc.org/upa_projects/voting_and_usability/workshop_2004*: "We need to consider not only voters, but everyone in the elections process: poll workers (and their materials and training), elections officials (and the usability of the tools they use to create a ballot)."

- ***Effective support for core election objectives***: Given the critical objectives of election accuracy, security, reliability, and ballot secrecy, does the documentation effectively educate and support election officials in managing election operations related to the VS so that high standards in each of these areas can be achieved?

In reviewing the user or operational manuals Diebold provides, these five criteria form the baseline for documentary usability evaluations. This report focuses on three subsets of election operations documents as exemplars of the Diebold documentation: those concerning GEMS; TSx touchscreen support, and optical scanning—both precinct count and central count.

5.2.1 Speed of Use

Does the documentation allow election officials to locate answers to their questions quickly?

Macro-Organizational Usability Considerations Diebold has generated a very large number of operating manuals, which it tends to divide into groupings it terms "suites." As concerns VS components, the suites are not simply a hardware manual and a software manual, or an election official's administrative operational manual and the maintenance/hardware manual. Instead, Diebold generally supplies a *multitude* of manuals. Yet when inventorying the manual suite in a given introduction, it is not unusual for Diebold to excerpt the list and exclude some manuals that provide valuable information not repeated elsewhere. The GEMS Reference Guide, for instance, is often not mentioned in the list of GEMS manuals at the front of a Manual introducing the system documentation.

Significant problems attend this proliferating approach to documentation. The logic of where to look for particular issues, explanations and information is not readily apparent. One can comb through several manuals without locating any information on the point in question. Second, instead of having only a few well-organized, well written manuals that election officials might begin to master as far as internal logic and content, it seems that the Diebold documentation effort has become one focusing on producing a quantity of manuals rather than their quality or overall usability.

For instance, the GEMS 1.18 operators' or users documentation includes:

- Election Administrator's Guide, Rev. 8.0 (543 pages).
- Product Overview Guide, Rev. 2.0, (13 pages).
- Reference Guide, Rev. 8.0 (355 pages).
- System Administrator's Guide, Rev 6.0 (95 pages).
- User's Guide, Rev. 12.0, (281 pages).
- JResult Client 1.1 User's Guide, Rev. 1.0, (28 pages).

In rough numbers, this is a total of over 1300 pages spread between six manuals. This team concludes that the documentation is unusable for reaching speedy answers.

At the beginning of several manuals Diebold presents the GEMS documentation in this manner:

> 1. *GEMS Election Administrator's Guide*, which provides a comprehensive description of all election management tasks involved in configuring an election using Diebold … election products, including the GEMS election management software and AccuVote-TS and AccuVote-OS voting and ballot counting devices.
>
> 2. *GEMS Product Overview Guide*, which provides an overview of Diebold … election products and product documentation.

3. *GEMS Reference Guide*, which provides a comprehensive description of the concepts involved in GEMS functionality.

4. *GEMS User's Guide*, which provides a comprehensive description of the implementation of GEMS functionality.

5. *GEMS System Administrator's Guide*, which provides a comprehensive description of all technical administrative activities involved in the usage of GEMS software and GEMS clients.

For county election managers, or anyone else who needs to become technically proficient in operating or managing GEMS, the documentation divisions as between the various manuals are not conceptualized well or presented accurately. Their division is particularly frustrating from the standpoint of operational management and trouble-shooting. Hence, a local official in charge of a particular managerial task (for instance, absentee ballots to be tallied on a central count optical scan system) would not find the instructions in one or two manuals but in parts of five separate GEMS manuals, and most likely needing to use three manuals repeatedly which range between 280-540 pages. She would also need to research the suite of AV-OS manuals, plus possibly the Digi-port manual and still others.

Not only are the manuals' division of information not logical from an operational standpoint, but the information needed from other manuals is not cross-referenced or well-indexed. A conscientious absentee ballot manager would need to generate his or her own manual in a cut-and-paste manner – if Diebold's effort to protect its manuals from photocopying did not interfere.

For Central Count AV-OS, Diebold documentation suggests this organization to its literature:

> AccuVote-OS device are described in the *AccuVote-OS Hardware Guide*, including the installation of AccuVote-OS Central Count firmware. For information on the configuration of the AccuFeed for use with the AccuVote-OS, refer to the *AccuFeed Hardware Guide*. For information on administrative considerations in the use of AccuVote-OS Central Count, refer to section 9 *Absentee Processing* the *GEMS 1.18 Election Administrator's Guide.*

Unfortunately, not all the information that is needed or relevant to setting up and managing AV-OS Central can be found in these three AV-OS manuals..

If an official sought the ***error message list*** for optical scanning, the listing would run numerous pages. But he would discover that it is not ordered alphabetically, by function, by election phase, or by any other discernible criterion. In the manual's text, the chapter 8 heading (which also is stated in the Table of Contents/Index) does not mention it as the Error Message chapter but instead lists it as the *Ballot Return Message*:

> **8. *Ballot Return Messages***
> This chapter lists all possible ***error messages*** that may occur in the course of ballot testing and counting. Each message is listed with the probable error cause as well as a recommended solution. Any ballot conforming to the ballot return criteria defined under the Reject Settings tab in the AccuVote-OS Options window in GEMS is returned by the AccuVote-OS. . . . *(emphasis added).*

This form of indexing requires that an official already know the particular terminology used by the vendor in order to discover critical information, Unfortunately, a large number of Diebold operational/user manuals lack an alphabetical index with information cross-listed using several key words, including vendor specific names, technical nomenclature and more common operational or election vernacular.

5.2.2 Accuracy and Consistency

Does the documentation provide one name for a component or process, or several conflicting answers? Are the operational facts of the equipment presented accurately?

Apparently, this vendor does not desire election official reliance on the accuracy of its manuals, as indicated by a disclaimer of warranty. Rather surprisingly, this vendor appears to include at the front of every operational or customer manual a warranty disclaimer. The most recent version was found in the *Readme* file that the vendor supplied with the portion of TDP files it submitted to the TTBR in mid-July. This 2007 *Readme* file may reflect the most recent version of the documentation disclaimer:

> **Disclaimer of Warranty**
>
> The ***information in this document is provided 'as is' and without warranty***. Diebold Election Systems will not be liable for any incidental, consequential, or other damages of any type or nature, resulting from the provision or use of the information contained herein. All information is subject to change at any time without notice. ***Users of this document assume sole responsibility for their use of the information contained herein, as well as any products, software, or other materials that may be provided by Diebold Election Systems***. Care should be exercised by such users to assure compliance with all applicable laws, rules, and regulations. *(emphasis added).*

Whether this language is unenforceable as a matter of California or other law is a question that the Secretary of State may want to explore. Non-lawyer election officials may assume language has been vetted and is legally enforceable since it appears in a document that presumably has been reviewed by certification authorities (though the enforceability of warranties would likely be well beyond the scope of certification reviews).

Examples of inconsistent and confusing parts nomenclature:

AV-OS: Precinct Count Optical Scan Hardware Guide: *Multiple, possibly inconsistent names for the ballot box compartments.* In addition to the "main" compartment, additional compartment names include "secondary," "alternate," "auxiliary," and "side," for a total of five names. While the documentation states there are four doors and one compartment has two doors, it does not state how many compartments are located in the scanner's ballot box. Some examples *(with emphasis added)*:

> Sorted ballots are dropped into the *secondary compartment* of the ballot box if the AccuVote-OS is installed in the ballot box . . . .
>
> **4.3.2. Separating ballots into the *alternate ballot box compartment***
> The AccuVote-OS may be programmed to sort ballots encountered any of the conditions selected. . . . Sorted ballots are dropped into the *secondary compartment* of the ballot box if the AccuVote-OS is installed in the ballot box….
>
> **3.2.7. Ballot box compartments**
> The *four doors* that access the ballot box compartments are locked using either of the ballot box keys. All ballot box compartments should be verified as being empty prior to voting begin. A small opening on the upper left side of the ballot box provides access to an *auxiliary compartment.* . . . The door on the lower left side of the ballot box also provides access to the *auxiliary compartment*, but this door should only be opened to remove ballots from the auxiliary compartment at the end of election day or to be counted on the AccuVote-OS. The back door of the ballot boxes is designed to allow the removal of ballots . . . or in case of an AccuVote-OS failure (where ballots are transferred to the *side compartment*).

3.2.9. Separating ballots in the ballot box
Ballots, such as write-in or blank ballots, may optionally drop into the *secondary compartment* within the ballot box.

Inconsistencies not only lead to miscommunications but also can be repeated in poll worker and other election official training.

5.2.3 Clarity

Is the documentation written unambiguously, so it is relatively easy to understand?

Speed in using manuals can depend not only on the quality of indexes but also on the clarity of the writing in the discussion desired. Unfortunately, most Diebold user manuals were pervaded with poorly edited, unnecessarily lengthy and confusing sentences. While election officials are rarely IT professionals, the manuals tended to rely on technical jargon that could easily detract from the larger election administrative or mechanical points that needed to be understood.

Virtually all of the manuals included one relatively well-edited Introduction. Each of these introductions appeared to have been reproduced from marketing information. Without exception, they touted the ease of use and high performance of the Diebold line of products. Had the balance of the manuals been edited to the same degree as these Introductions, the clarity and usefulness of the writing would have improved. For all manuals other than the AV-OS Hardware manual, this introductory passage seemed to be the only portion of the writing that was edited for clarity and persuasion. The contrast between the introductory overview and the operating instructions and explanations seems quite sharp and ultimately detrimental to the effectiveness of the manuals

Examples of well written passages or manuals:

- The AV-OS Hardware Manual is a relatively well written resource, with reasonably good internal organization and clear writing.
- The GEMS 1.18 Reference Guide includes some very well written definitions in its Appendix A: Glossary, that avoid dependence on using the word sought to be defined:

  Closed primary: A primary election in which voters vote on ballots containing only races corresponding to their political party as well as any non-partisan races in the election.

  Export: The composition of election and election results information into an ASCII file format.

  Election Summary report: A report summarizing election results by race, across the entire jurisdiction, presented according to customization criteria.

Most other manuals have significant problems with language precision and lack of clarity; they**:**

- provide definitions with dependence on using the word sought to be defined: [36]

  Monitor Script: A JResult Client configuration, defined in terms Monitor Script Properties, in turn defined in terms of sets of reporting sets and precincts or districts.
  Monitor Script Properties: Monitor Scripts are defined in terms of Monitor Script Properties, which

[36] 2005 GEMS 1.18 Reference Guide, Glossary.

are in turn defined in terms of sets of reporting sets and precincts or districts

- provide confusing and imprecise definitions that fail to clarify the differences between related but distinct words:

> ***Ballot*:** A Ballot refers to a rotated ballot style. A single card comprises all language variants for the card.
> ***Card***: A rotated, physical document containing a unique set of races. One or more cards comprise a single card style. A single card comprises all language variants for the card.
> ***Card Style***: A physical document that is contained within a ballot style; a ballot style may contain one or more card styles. A single card style comprises all language variants for the card style.
> ***Deck***: A set of ballots processed in Central Count, delimited by Batch Header or Batch Start cards.

The Glossary provides no entry for ***"batch"*** although the documentation sometimes uses the words "deck" and batch interchangeably, and other times with apparently distinct meanings. The usage sometimes appears to suggest that a *"**deck"*** is the electronic representation of a physical ***"batch"*** of ballots processed in the central count scanning system, but the documentation lacks consistency in usage, and the definition provided here does not support that distinction.

- miss an opportunity for promoting mechanisms for checking tabulation work with an aim for achieving greater accuracy

> ***Ballot Audit***: A function allowing the review of ballots uploaded from AccuVote-TS units.

Ballot audits can occur with optical scanning processes as well, and arguably should occur with all tabulation equipment.

Operational user manuals display problems with clarity, precision and jargon throughout. A few examples include:

> **AV-OS Precinct Count**[37]
>
> **3.2. Ballot tallying** *[describing how AccuVote-OS ballots are counted and tallied]:*
> For every ballot encountered, the Times Counted counter is incremented for the ballot as well as the Times Write-In counter if the ballot contains at least one write-in candidate selection in a non-overvoted race.
>
> **3.3.5. Shadow races**
> A race may be configured to automatically tally to reporting districts that intersect the district over which the race runs, or to voter groups with which partisan ballots in which the race runs are affiliated. A race is defined with Type Shadow with district, voter group, and all other parameters of a candidacy, with as many Shadowed races linked to the Shadow race as necessary to satisfy reporting requirements
>
> **3.5. Programmable ballot sort conditions**
> *[AccuVote-OS can be programmed to sort ballots]* encountering any of the conditions selected under the Sort Ballots With column under the Reject Settings tab in the AccuVote-OS Options window in GEMS.
> Sorted ballots are dropped into the secondary compartment of the ballot box if the AccuVote-OS is installed in the ballot box, otherwise, if ballots are being counted in batch mode with the AccuFeed installed, a sorted ballot is dropped into the AccuFeed outfeed tray, a message is displayed on the

[37] 1.96 User's Guide (2005).

AccuVote-OS LCD indicating that a sort condition has arisen, alternating with a prompt to press the Yes button and continue.

7.8.2. Tallying ballots with exception conditions
In section *7.8 Ballot rejection*, you are presented with information pertaining to the processing of ballots that are returned as a result of meeting certain ballot return conditions programmed in GEMS. This section explains in simple terms how ballot tallying is affected ballots with any of these conditions are accepted by the AccuVote-OS, either because the election has not been configured to return ballots with the said return condition, or ballots with this condition have been encountered, but fed into the AccuVote-OS in override mode.

Other problematic examples from the same precinct count AV-OS manual include:

The Times Counted counter is incremented for every race on the ballot, as well as the Total Votes counter for every candidate and write-in position selected in every nonovervoted race.

For every open primary ballot with overvoted crossover races encountered, the Times Counted and Times OverVoted counters are incremented for the ballot as well as the Times Write-In counter if the ballot contains at least one write-in candidate selection in a non-overvoted, nonpartisan race

The AccuVote-OS may be programmed to return ballots encountered any of the following conditions selected under the Return Ballots With column under the Reject Settings tab in the AccuVote-OS Options window in GEMS.

Central Count Optical Scan manuals fail to distinguish between the batch and the deck**:**

Batch numbers are pre-assigned from *Batch Header* cards, where GEMS assigns the batch number coded onto the Batch Header card as it is fed. It is not possible ... to assign a Batch Start card which has already been used unless the original *batch* is deleted from the database. The AccuVote Ender card is used to end a batch, at which point the batch is committed to the GEMS database. If the counting of a batch is interrupted (ie. the connection to the host computer is lost) the contents of the *deck* are also lost.
[More reasons than this... Ed.]

Frequently, the documentation fails to use common titles for events or conditions, rendering the user unable to effectively use indices or lists of issues:

10.2. Equipment *recuperation*
If the AccuVote-OS or memory card malfunctions and requires replacement. . . . .

5.2.4 Risks

Where a given procedure holds risks to a successful election if not performed precisely correctly, does the documentation flag the point with visual cues and exceptionally clear writing? Is contingency planning discussed adequately?

This team would suggest that the vendor accorded insufficient attention to the risks potentially accruing to election employees personally and individually if the VS vendors omit disclosures of risks to quality election task performance. Additionally, we believe that the documentation should augment election workers' knowledge of how to protect themselves from the possibility of rogue employees attempting to leave evidentiary trails suggesting that another worker had engaged in wrongful conduct.

These risks and the mitigation strategies should be disclosed for employee protection.

A clearly written notification of a major precinct count optical scanning risk is stated here with practical advice designed to mitigate the risk:

> **7.6. Replacing a full ballot box**
> Each ballot box holds up to 1500 ballots in each of the main compartments. Spare ballot boxes should be provided to polling places anticipating heavy voting. We recommend checking the contents of the ballot box when the ballot counter reaches 1000, and using the following procedure to replace the ballot box once it is full.

Unfortunately, the note does not include a warning flag or other distinctive visual cue. The manual presents a photograph, however, that does assists in visualizing the situation and point to recall.

Overall, the Diebold manuals are generally bereft of warning boxes, special flags or colors for dangers and risk factors, or other techniques for emphasis. Many critical risk factors are not mentioned, or are understated, buried in the midst of long passages of text.

The documentation inadequately addressed some serious risks or did not disclose them at all:

- ***Election results database growth.*** GEMS use of the JET engine means the database is limited to 2 gigabytes; database corruption is a well documented risk thereafter.[38] Given the risks to election database integrity, all factors that can cause rapid database growth should be disclosed with appropriate mitigation strategies. Currently, the Diebold optical scanning manuals we reviewed did not even advise that the GEMS database file (containing size information) should be monitored during central count scanning.
- ***Central count scanning.*** The scanning operations can sometimes result in defective data being uploaded into GEMS. The GEMS manuals include some instructions for recognizing and monitoring for the need for a batch/deck deletion, but they fail to note the serious risks for election data from operator errors, and omit needed mitigation strategies. The documentation also sidesteps offering careful instructions on how to guard against duplicate scannings of a ballot batch and failures to re-scan a deleted batch.[39] To the degree that different types of batch start cards can be used for this internal auditing and tracking of batches, improved election accuracy can result. Thus, these and other strategies should not only be disclosed but effectively taught via the documentation. Additional strategies to improve accuracy and avoid risks could involve instructions on using scanning audit logs effectively and creating backup data points by periodically burning database records during scanning and other ballot tabulations.
- ***Presence and Use of Modems.*** In both the TSx and AV-OS, modems generate certain risks to election results and to the reliability of the VS equipment, as discussed in the Diebold Red Team and Source Code reports.
- ***Audit Log information****:* These logs offer substantial value to security and accuracy values, but unless the documentation explains how to use them and the various purposes for which they are relevant, their presence is nearly meaningless. Their value extends from discerning some types of tampering, to identifying system failures or problems and operator errors, to trouble-shooting and improving election operational performance.

---

[38] *See, e.g.,* Diebold Source Code Team report.

[39] The Cuyahoga Collaborative Public Audit discovered deleted ballot batches that had not been rescanned, and batches that had been scanned more than once. The Final Report is located at www.csuohio.edu/cei/.

- ***Networks***: The Diebold documentation made available for this review sidestepped the risks of data dregradation, the role and specifications of appropriate network stress testing, and major risks to the GEMS server created by the network connections.
- C***oncurrency issues and limitation.*** JET presents the possibility of concurrency problems that can lead to data degradation or system failures. Diebold documentation remains mute on this matter.
- ***Disclosures of known VS vulnerabilities.*** Published studies and election administrative experience has developed a set of vulnerabilities for various components of the Diebold VS. Operational mitigation strategies then become key protections for election integrity. But this vendor fails to disclose known vulnerabilities, and fails to offer appropriate mitigation strategies.

5.2.4.1 Contingency Planning

Step-by-step procedures covering election administration under optimal circumstances are essential. Equally essential are detailed strategies for handling problems that arise. The Diebold customer documentation offered helpful strategies for dealing with contingencies such as responding to unexpected polling place supply shortages on Election Day, cold-swapping voting device batteries in response to isolated power failures, and physical transportation of memory cards when remote upload of voting data fails. In areas in which the voting systems themselves appear to be malfunctioning, or when possible tampering is detected, however, this vendor's documentation offers few prudent and realistic instructions for troubleshooting, mitigating, recovering and reporting.

In order to effectively handle adverse conditions, election officials need precise information on the state of the voting system, on its recovery features and on the level of fault tolerance that the system reliably exhibits in specific situations. The Diebold documentation reviewed for the TTBR provides little guidance to election administration officials on recovering from adverse events and mitigating the impact of failures, errors and malicious actions. The response, recovery and mitigation instructions presented in the documentation are often inappropriate, incomplete, obvious or impractical.

As detailed below, several contingency response procedures in the documentation rest on dubious assumptions about the certainty with which one can determine the precise state of system components in the aftermath of system failures and other adverse events. Such assumptions may lead to responses that can cause the situation to deteriorate by directing that election operations proceed without verifying that executable code and data structures are intact following a system failure or security breach. Some of the vendor's proposed contingency plans appear to risk propagation of malicious code.

In cases where malicious intrusion into voting systems is suspected, the manuals offer little information about the need to isolate equipment to contain the threat and preserve a device intact for forensic examination. The manuals offer no guidance whatever on what general sorts of forensic investigations would be appropriate.

Dubious Contingency Plans The following examples illustrate the wide range of questionable contingency response information contained in Diebold's customer documentation. They make categorical assertions of questionable technical accuracy or prudence, instructing workers to proceed with potentially damaged or compromised equipment, or reassuring workers about the reliability of the system state.

- "The technician should replace the hard drive and attempt to extract the contents of the failed hard drive to the replacement drive. If this restoration activity is successful, then the integrity of the

GEMS installation and election configuration should in no way have been compromised."[40]

- "The AccuVote-TS cannot be voted or rendered functional in a meaningful way unless a programmed memory card is installed." [41]

- "Vote totals cannot be accumulated to an AccuVote-OS memory card other than with the availability of AccuVote-OS ballots. It is possible that an unauthorized party manufacture *(sic)* ballots, but without specific knowledge of ballot specifications and electronic ballot images generated by the GEMS election management software, it would be unlikely."[42]
- "If ballots are available to the unauthorized party, and ballots are marked, it will not be possible to tally results without the availability of an AccuVote Ender card. If an AccuVote Ender card is present, results may then be tallied on the memory card."[43]

- "If power failure results in damage to the voting machine as a result of a lightning strike, then the voting machine must be replaced. In case of an either an AccuVote-TS or AccuVote-OS unit, the memory card should be removed from the damaged unit and installed in a replacement unit, and voting continued."[44]

- "Erroneous configuration or behavior should result in the immediate replacement of the voting device. [...]Since the tallies of official ballots cast on the device are resident on the memory card, the seal number should be recorded of the failed unit, the memory card removed from the device, the memory card then installed in the replacement unit, a new seal number applied, and recorded."[45]

- "If a voting machine is damaged at a polling location, then the voting machine must be replaced. In case of an either an AccuVote-TS or AccuVote-OS unit, the memory card should be removed from the damaged unit and installed in a replacement unit, and voting continued, provided that the memory card has remained intact."[46]

From a technical standpoint, some guidance the manuals offer is reasonable, but by no means complete. For instance, Diebold recommends responses for events where evidence of tampering or system failure has arisen, but these instructions are missing important additional steps. Examples include:

- "If unauthorized software is detected on the GEMS server, that software should be removed immediately. Software uninstallation should be performed by qualified IT staff, upon approval by Diebold Election Systems, Inc."[47]

- "In the event that unauthorized physical access to the GEMS server is detected, either as a result of observing an attempt at physical access or by reviewing the operating system audit trail, it is essential that physical security be enhanced in the environment of the GEMS server. Physical security should be enhanced at least for the duration of the election lifecycle."[48]

---

40 *GEMS 1.18 Election Administrator's Guide, Revision 8.0*, p. 16-59.
41 *GEMS 1.18 Election Administrator's Guide, Revision 8.0,* p. 16-61.
42 *GEMS 1.18 Election Administrator's Guide, Revision 8.0*, p. 16-62.
43 *GEMS 1.18 Election Administrator's Guide, Revision 8.0,* p. 16-62.
44 *GEMS 1.18 Election Administrator's Guide, Revision 8.0,* p. 16-66.
45 *GEMS 1.18 Election Administrator's Guide, Revision 8.0,* p. 16-60; statement is repeated in reference to the AccuVote-OS on page 16-62.
46 *GEMS 1.18 Election Administrator's Guide, Revision 8.0,* p. 16-65.
47 *GEMS 1.18 Election Administrator's Guide, Revision 8.0*, p. 16-59.
48 *GEMS 1.18 Election Administrator's Guide, Revision 8.0,* p. 16-58.

Other contingency instructions are obvious or unneeded, for example:

- "If power failure results in damage to the GEMS server as a result of a lightning strike, then the appropriate components of the GEMS server must be recuperated."[49]

- "Damaged Voter Card Encoder units that are no longer operational should be replaced with replacement Voter Card Encoder units."[50]

- "In the event that a critical employee in the election management process becomes suddenly unavailable, it is essential that an alternate qualified employee be located and trained immediately. [...] Initially, a replacement employee occupying a critical position should be provided as much assistance as possible from related staff positions."[51]

- "If power failure occurs during election closing, memory cards should be driven in to the election administration office in place of attempting to upload memory cards."[52]

- "AccuVote-OS units should be monitored in the course of election day in order to ensure their continued ability to count ballots."[53]

- "In the event of delay of results accumulation to the GEMS server, an announcement should be made to the public pertaining to the delay of results pending resumption of power."[54]

Some contingency response plans are impractical or legally impermissible. For example:

- "If memory cards have been lost, the election must be re-scheduled."[55]

- "If the AccuVote-TS unit with installed memory card disappeared from a voting location in the course of or at the conclusion of election day, but prior to uploading, then all votes on the machine will be lost, and all voters at the polling location will have to be called in to re-vote. All machines present at the polling location may upload results at the end of election day, barring the missing unit, and once all voters at the polling location have completed the repeat vote, the new results are uploaded to the GEMS server after the original results for the vote center are cleared in the GEMS database."[56]

- "If both the AccuVote-TS unit and installed memory card are damaged in the course of or at the conclusion of election day, but prior to uploading, then all votes on the machine will be lost, and all voters at the polling location will have to be called in to re-vote."[57]

---

49 *GEMS 1.18 Election Administrator's Guide, Revision 8.0*, p. 16-57.
50 *GEMS 1.18 Election Administrator's Guide, Revision 8.0,* p. 16-64.
51 *GEMS 1.18 Election Administrator's Guide, Revision 8.0*, p. 16-57.
52 *GEMS 1.18 Election Administrator's Guide, Revision 8.0,* p. 16-66.
53 *GEMS 1.18 Election Administrator's Guide, Revision 8.0,* p. 16-62.
54 *GEMS 1.18 Election Administrator's Guide, Revision 8.0*, p. 16-57.
55 *GEMS 1.18 Election Administrator's Guide, Revision 8.0,* p. 16-56.
56 *GEMS 1.18 Election Administrator's Guide, Revision 8.0*, p. 16-61.
57 *GEMS 1.18 Election Administrator's Guide, Revision 8.0,* p. 16-65; statement is repeated in reference to the AccuVote-OS on page 16-66.

5.2.5 Effective support for core election objectives

Given the critical objectives of election accuracy, security, reliability, and ballot secrecy, does the documentation effectively educate and support election officials in managing election operations related to the VS so that high standards in each of these areas can be achieved?

5.2.5.1 Poll Worker Support

Diebold documentation does not include a dedicated poll worker guide for either type of polling location voting equipment. Rather, it includes manuals designed for election officials to use in constructing a poll worker guide. These vendor guides to creating a local poll worker manual are deficient in numerous ways. One key reason this is not an acceptable vendor approach for achieving quality local election support is that local election officials are largely unfamiliar with usability criteria for manuals. It would be an unusual election official who had experience writing effective manuals about complex technical matters for a non-technical audience of various literacy levels. Thus, self-sufficiency in developing poll worker manuals or training would be illusive.

A quality ***template*** (with an unlimited, no fee license) for a poll worker manual would seem to be a reasonable expectation of each VS vendor. The Diebold poll worker manuals cannot and do not function as such a template. Each of them is poorly organized; include a range of technical information not suitable or needed for the poll workers; are not clearly written; fail to stress or even explain the security role of the poll workers effectively; and seem obviously not developed by anyone who has experience in teaching poll workers. These manuals cannot function as quality written materials in support of teaching election workers.

5.2.5.2 Accuracy and Verifiability of Election Results

Some GEMS programming options that are detailed in the manuals can promote voter accuracy and election administrative improvements -- if they were more widely known and used effectively:

> **GEMS** *[Programming options for optical scanning]*
>
> **7.8.1. Programmed ballot return condition:** to return ballots according to special voting "conditions." The options are:
> 1. Overvoted ballots
> 2. Undervoted ballots
> 3. Blank voted races
> 4. Blank voted ballots
> 5. Overvoted crossover races
> 6. Overvoted straight party races

> **GEMS** *[maintains tallies or "counters" not widely known that can promote auditing and improvements]:*
>
> **Counters**
> The following counters are maintained in the process of counting and tallying AccuVote-OS ballots. For each ballot:
> 1. ***Times Counted*** represents the total number of cards counted, and is incremented for every card counted
> 2. ***Times Blank Voted*** represents the total number of cards with blank voted races, and is incremented for every card with one or more blank voted races.

Ballot Processing
3. ***Times OverVoted*** represents the total number of cards with overvoted races, and is incremented for every card with one or more overvoted races.
4. ***Times UnderVoted*** represents the total number of cards with under voted races, and is incremented for every card with one or more undervoted races.
5***. Times Write-In*** represents the total number of cards with write-in votes, and is incremented for every card with one or more write-in candidate selections.

For each race:
1. ***Total Votes*** is maintained for every candidate and write-in candidate position, and is incremented for every valid vote assigned to the candidate or corresponding write-in position
2. ***Times Counted*** represents the total number of times the race was counted, and is incremented every time the race was encountered
3. ***Times Blank Voted*** represents the total number of times a blank voted race was counted, and is incremented every time a blank voted race was encountered
4. ***Times OverVoted*** represents the total number of times an overvoted race was counted….

A misused Precinct Header card can disrupt election accuracy and is insufficiently flagged.

5.2.5.3 Reliability

The rate of failures in technical equipment is often referred to as its "reliability." This point is crucial for VS as failures in a component can disenfranchise voters or disrupt an election. Metrics for component failure rates are not provided in the documentation, and it appears the vendor does not suggest that the jurisdiction keep failure records of components.

Reliability issues can be more effectively managed if full disclosure of risk factors occurs. Some key risks that affect the reliability of the VS are discussed in section 5.2.4 above.

The Diebold VS Hardware manuals often include sound suggestions and guidance for maintenance issues that can affect the reliability of VS components. .

Poll worker guides are possibly at their strongest in the diagrams and instructions on how to set up and close down voting devices, which also relates to VS component reliability.

5.2.5.4 Ballot Secrecy

We examined the vendor documentation for indications that the vendor was effective in identifying ways to promote ballot secrecy, including by flagging risk points for compromising a voter's interest in maintaining the confidentiality of his or her ballot choices. The discussion that follows leaves to the other Diebold teams an assessment of the degree to which a voter's selections may be traced back to a particular individual via technical means and instead focus on the ballot secrecy provided to the voter for period during which the ballot is being marked and cast.

AccuVote-TSx The AccuVote-TSx unit has two plastic flaps or doors that swing away from the front of the device to create visual barriers to the left and right of the touchscreen where the visual ballot is displayed. These are intended to reduce the visibility of the touchscreen to observers on either side of the voter. The primary visual barrier protecting the front of the screen is the voter's body.

The documentation refers to the doors (called "privacy panels") as the safeguard against observers

seeing a voter's votes as they are cast.[58] The materials do not discuss ways to improve or promote the privacy doors' use for better assuring privacy. Further, one critical consideration for TSx ballot privacy -- physical arrangement of the machines in the polling place – is accorded cursory attention, with only a suggestion that the local officials provide a physical layout.

The documentation does highlight an important feature available to voters who select an audio ballot. These voters are given the option of having the touchscreen remain entirely blank as they make their selections. This feature clearly augments the ballot secrecy of the audio-only ballot.

In addition to the documentation's omission of privacy issues that arise for the visual ballot voter that are not addressed by the privacy panels, it further overlooks the role that small physical size can play in the voter's ballot being seen as it is being marked, or the choice of larger typeface, or the angle that the screen is set. The documentation does not offer any guidance on the degree to which magnified ballots might require additional steps to achieve ballot secrecy. Clearly, a voter should not have to make a trade-off between ballot legibility and ballot secrecy.

Many problems of the voter's selections being observable may be mitigated by careful physical configuration of the machines or the introduction of additional, freestanding visual barriers. An election administrator or poll worker attempting to increase the privacy of voters using the AccuVote-TSx at a polling place, however, could not rely on the documentation provided with the voting devices for guidance. Such problems may be aggravated during early voting, when small numbers of voting devices may be located in facilities housing other public activities, and which are not configured exclusively for voting.

Other documentation that might be helpful in advancing voter secrecy when using the AccuVote-TSx would be instructions on how a poll worker may assist a voter with a question without viewing the touchscreen. A mechanism by which a voter could temporarily blank the screen while seeking aid from a poll worker might also be of value.

One statement in the Diebold customer documentation gives an indication of its overall treatment of the ballot secrecy issue. In discussing measures to take when there is a concern that someone has been able to manufacture multiple, fraudulent voter access cards, the *GEMS 1.18 Election Administrator's Guide* states:

> *AccuVote-TS units at the polling place may be configured in a manner that assures voter privacy, but allows poll workers to detect attempts made to perform repeat insertions of voter access cards into the AccuVote-TS smart card reader*.[59]

The documentation does not offer any suggestion as to how this might be done. It simply states that it is possible to set the DREs up so as to observe the insertion of smartcards, and that it should be done in certain situations. Encouraging poll workers to keep an eye on voters activities, somehow without observing their votes or otherwise intimidating them, to address a supposed security breach may do more to compromise ballot privacy than to augment security.

Any testing-based assessment of ballot secrecy is complicated by the difficulty in quantifying degrees of "ballot secrecy" with reference to the human observer. The Wyle report on the AccuVote-TSx does not describe any testing conducted to determine observer proximity and viewing angles that permit a voter's

---

[58] "The AccuVote-TSX privacy panels are maintained in an open position on election day, in order to protect the voters' privacy in the course of voting." *AccuVote-TSx Hardware Guide, Revision 8.0,* p.2-1.

[59] *GEMS 1.18 Election Administrator's Guide, Revision 8.0,* p. 16-64.

selections to be seen, or give details of the physical configurations that were found to provide adequate ballot secrecy. The nearly limitless array of venues in which voting takes place also hinders standardization and quantization of any testing methodology.

Accu-Vote-OS Assuming that voters are afforded reasonable privacy for filling-out paper ballots, the degree of ballot secrecy depends on the support afforded voters using precinct-based Accu-Vote-OS devices – particularly when the voter places the ballot into the scanner. The documentation discusses the use of opaque sleeves to cover the paper ballot while it is being fed into the AV-OS.[60] It is less effective in describing the poll worker's procedures to maintain ballot secrecy if the scanner jams or returns the ballot. The poll worker manual for AV-OS should stress these procedures and the importance of not observing a voter's selections, such as not removing the sleeve before feeding the ballot. Voter privacy would be advanced by clear warnings of these risks in the documentation, and by encouraging the creation of local polling place policies or mechanisms to reduce the opportunities to view a returned ballot, including from a line of waiting voters.

60 "Secrecy sleeves are used to cover the marked contents of the ballot while the ballot is fed into the AccuVote-OS, maintaining the secrecy of the voter's selections on the ballot." *AccuVote-OS Hardware Guide, Revision 5.0*, p. 2-4.

CHAPTER 6

Security

We evaluated the security policies presented in Diebold's customer documentation, their internal documentation and the documentation submitted to ITAs as part of the Technical Data Packages with which we were provided. While there are elements of sound policies throughout, we found several significant problems, including:

- Inconsistent security policies in different documentation sets.
- Problematic statements and assurances in the documentation distributed to customers.
- Failure to implement reasonable and consistent security configurations for systems delivered to customers.

6.1 Inconsistent Security Policies

Among the information that the VSS requires as part of the confidential Technical Data Package (TDP) that the vendors submit to testing authorities during the qualification testing process are "mandatory administrative procedures for effective system security."[61] This information is crucial to assessing the security of a voting system because security vulnerabilities have as much to do with how systems are used and administered as with the systems themselves. A highly-secure system may be subject to all manner of attack if it is not operated according to sound usage policies. Likewise, risks due to holes in a system's security may be mitigated by strict adherence to well-designed usage policies.

Diebold submitted extensive materials labeled as *Appendix X* as part of its GEMS TDP.[62] The *System Functionality Description* section of the GEMS TDP states that the mandatory administrative procedures for GEMS are contained in TDP *Appendix X* and in the *GEMS 1.18 Election Administrator's Guide*.[63] The *Election Administrator's Guide,* however, is distributed to customers as part of the GEMS documentation set, while *Appendix X* is part of the confidential TDP that customers do not receive. No document describing a client security policy similar to that in *Appendix X* was included in the customer documentation submitted by Diebold to the TTBR, nor is any mention made in the customer documentation to the existence of such a policy.

In assessing the adequacy and accuracy of security information in both the GEMS TDP and the GEMS customer documentation, many discrepancies surfaced. A great number of security regulations are substantially different as between the two Diebold policy documents. The mandatory policy set forth in

61 2002 VSS @ 2.3.2.1 (g).

62 *GEMS 1.18 TDP Appendix X: Client Security Policy*.

63 *GEMS 1.18 Technical Data Package: System Functionality Description, Revison 3.0,* p. 2-5.

areas such as user account management, password security, system auditing and physical security is markedly different in *Appendix X* and in the GEMS customer documentation. Generally, the policy in *Appendix X* is quite a bit stricter (and more secure) than that disseminated to the electoral jurisdictions in the GEMS documentation..

This set of discrepancies raises serious concerns. First, the qualification and certification processes are premised on the recommendation of the Independent Testing Authorities who evaluate the systems, including system security. An ITA's findings and recommendations are founded on both its own testing and the information in the Technical Data Package supplied by the vendor. The ITAs' determinations that Diebold's systems met or exceed the security requirements of the VSS are likely be predicated, to a significant degree, on the security policies that they expected to be in place in the counties in which the machines are deployed.

The fact that a different, less stringent set of security policies is contained in the customer documentation makes it possible that the ITAs' conclusions were based on the reasonable but flawed assumption that the *Appendix X* security policy would be reflected in the customer documentation and would be in place when the systems were used. Not only is the *Appendix X* policy absent from the customer documentation Diebold submitted to the TTBR, there is no indication in any of the customer documentation that a different policy was submitted for testing. The following tables present a comparison between the mandatory security policy distributed to customers and that submitted to ITAs.

Tables 6.1 – 6.10 Security Policy Discrepancies

The tables below list the provisions of two election systems security policies created by Diebold for use with its voting systems. The columns on the left show the security policy contained in the documentation Diebold provides to customers, the columns on the right show the mandatory client security policy Diebold submitted for ITA testing as part of the GEMS Technical Data Package.

See also Tables 6.11 – 6.14 for details of the security configuration of the GEMS server provided by Diebold for the TTBR study. Note the variance between the user account, password and auditing policies documented by Diebold in Tables 6.1 – 6.10 and those actually implemented on a delivered system.

Table 6.1 System Account Policy Discrepancies

System Accounts	
Diebold Security Policies Distributed to Customers Source: *GEMS 1.18 Election Administrator's Guide, Revision 8.0*, unless otherwise indicated.	**Mandatory Client Security Policy Submitted to ITA** Source: *GEMS 1.18 TDP Appendix X: Client Security Policy*.
System is initially configured with anonymous accounts. All system user IDs and passwords must be maintained in a private and secure manner, conforming with local security requirements.	Every user must have a single unique user-ID and a personal secret password. This user-ID and password will be required for access to multi-user computers and computer networks. Each computer and communication system user-ID must be unique and forever connected solely with the user to whom it was assigned.

The election Security Administrator should be assigned responsibility for the issuance and renewal of user IDs and passwords.	Without specific written approval from Director of Information Security or equivalent title and authority, administrators must not grant system privileges to any user. User-IDs may be granted to specific users only when approved in advance by the user's immediate supervisor. Prior to being granted to users, business application system privileges must be approved by the involved information owner.
Accuvote-TSX: There are three different types of access cards: voter access cards, Supervisor cards, and Central Administrator cards. (*AccuVote-TSX Pollworker's Guide v. 5.0*, page 5-1) Voter access cards are encoded by pollworkers, given to voters, used once, and then returned to the pollworkers. Once a ballot has been cast with a voter access card, the card is deactivated and needs to be re-encoded with ballot information before it can be reused. (*AccuVote-TSX Pollworker's Guide v. 5.0*, page 5-1) Supervisor cards are given to designated pollworkers and used to exit the Official Election screen and access the pollworker functions, such as encoding voter access cards. Supervisor cards are also used to end voting on the AccuVote-TSX unit at the end of an election. (*AccuVote-TSX Pollworker's Guide v. 5.0*, page 5-1) Central Administrator cards can be used by election administrators to access the administrative menu interface. They may also be used in lieu of a pollworker card to access pollworker functions, but it is not recommended that Administrator cards be distributed to pollworkers. Both Supervisor cards and voter access cards are required on election day. (*AccuVote-TSX Pollworker's Guide v. 5.0*, page 5-1) The same Supervisor password must be used for all memory cards in an election, but the password may vary from election to election. (*AccuVote-TSX Pollworker's Guide v. 5.0*, page 5-1)	The system privileges granted to every user must be reevaluated by the user's immediate manager every six (6) months. This reevaluation involves a determination whether currently-enabled system privileges are still needed to perform the user's current job duties. All user-IDs must automatically have the associated privileges revoked after a thirty (30) day period of inactivity. Management must promptly report all significant changes in end-user duties and/or employment status to the system security administrators handling the user-IDs of the affected persons. So that their privileges may be expediently revoked on short notice, records reflecting all the computer systems on which users have user-IDs must be kept up-to-date. All information systems privileges must be promptly terminated at the time that a worker ceases to provide services to the Diebold Election Systems client jurisdiction. Unless approved by authorized officials at the client jurisdiction and Diebold Election Systems, unauthorized staff and contractors must not enable any trusted host relationships between computers connected to the network containing product servers, workstations, or voting and ballot counting devices. A trusted host relationship involves the sharing of data files or applications across computers, or the elimination of the need to log-into more than one computer.

Table 6.2 **Account Privileges and Access Limitations Policy Discrepancies**

Account Privileges and Access Limitations	
Diebold Security Policies Distributed to Customers Source: *GEMS 1.18 Election Administrator's Guide, Revision 8.0*, unless otherwise indicated.	**Mandatory Client Security Policy Submitted to ITA** Source: *GEMS 1.18 TDP Appendix X: Client Security Policy*.
User IDs and passwords are used to restrict access to software functions to authorized individuals only. Access to user IDs and passwords should be limited to authorized users of the corresponding application only. There are two access security levels in GEMS: • Administrator – may perform any activity in GEMS • Non-Administrator – prevented from changing the election status once the election status has been set to 'Set for Election' and from clearing vote counters <u>Accuvote</u>: A Supervisor card must be entered into the smart card reader, followed by a Supervisor password in order to end the election or access to critical administrative functions on the AccuVote-TS or AccuVote-OS. Access to voter access card encoding devices, such as Voter Card Encoder or VCProgrammer, is restricted to authorized pollworkers only. A single account is used by all pollworkers dialing logging in to the GEMS server to upload memory cards. The smart card key, the data key, and the Supervisor password should be changed across all smart cards-activated devices used in the election. Limit Access to AccuVote-OS Ender Cards Access to the Supervisor card and Supervisor password should be limited to the chief election judge at the polling	Beyond that which they need to do their jobs, computer operations staff must not be given access to—or permitted to modify--production data, production programs, or the operating system. Privileges must be established such that system users are not able to modify production data in an unrestricted manner. Users may only modify production data in predefined ways that preserve or enhance its integrity. In other words, users must be permitted to modify production data ONLY when employing a controlled process approved by management. System privileges must be defined so that non-production staff (internal auditors, information security administrators, programmers, computer operators, etc.) are not permitted to update "production" election system-related information. Special system privileges must be granted only to those who have attended an approved systems administrator training class. The number of privileged user-IDs must be strictly limited to those individuals who absolutely must have such privileges for authorized business purposes. System privileges beyond the capabilities routinely granted to general users must be approved in advance by the Information Security Manager. All software installed on multi-user systems must be regulated by approved access control systems software. This means that a user's session must initially be controlled by approved access control systems software, and if defined permissions then allow it, control will then be passed to separate application software.

locations. Equipped with the Supervisor card and password, the chief election judge has exclusive jurisdiction to end the election on election day.	Multi-user systems administrators must have at least two user-IDs. One of these user-IDs must provide privileged access and be logged; the other must be a normal user-ID for the day-to-day work of an ordinary user.

Table 6.3 **Password Policy Discrepancies**

Password Policies	
Diebold Security Policies Distributed to Customers Source: *GEMS 1.18 Election Administrator's Guide, Revision 8.0*, unless otherwise indicated.	**Mandatory Client Security Policy Submitted to ITA** Source: *GEMS 1.18 TDP Appendix X: Client Security Policy*.
Passwords assigned should be sufficiently difficult to guess so that users are not tempted to impersonate other users. Network and operating system password features usually support mechanisms for preventing the use of trivial passwords Between elections, customers should review all defined users and change all passwords to reduce the exposure to password guessing. All security related data, including security keys, must be maintained confidential at all times.	Passwords must be at least 7 characters. Systems should force users to change their passwords at least every 45 days. On all multi-user machines, system software or locally developed software must be used to maintain an encrypted history of previous fixed passwords. The history file must minimally contain the last ten (10) passwords for each user-ID.
Passwords are set by administrative users which may use a password generator if deemed appropriate.	If passwords or Personal Identification Numbers (PINs) are generated by a computer system, all software and files containing formulas, algorithms, and other specifics of the process must be controlled with the most stringent security measures supported by the involved computer system.

Table 6.4 **Configuration Security Discrepancies**

Configuration Security	
Diebold Security Policies Distributed to Customers Source: *GEMS 1.18 Election Administrator's Guide, Revision 8.0*, unless otherwise indicated.	**Mandatory Client Security Policy Submitted to ITA** Source: *GEMS 1.18 TDP Appendix X: Client Security Policy*.
No software should be installed on the GEMS server or any other PC installed with election management software other than the approved software, as and when authorized, and performed by an authorized official only. No Diebold Election Systems, Inc. software product should be used in a manner other than as intended, that is, in a manner contravening recommended usage procedures provided in Diebold Election Systems, Inc. product support documentation. No alteration of system files should be performed on the GEMS or any other PC installed with election management software, unless explicitly approved by authorized officials, and in an authorized manner only. The versions of all software products used in an election should be verified prior to an election, and verified again prior to critical points in the election management process.	Unless permission of the Director of Information Security has been obtained, the use of direct database access utilities in the production environment is not permitted because these programs will circumvent database synchronization and replication routines, input error checking routines, and other important control measures.

Table 6.6 **Network Connectivity and Security Discrepancies**

Network Connectivity and Security	
Diebold Security Policies Distributed to Customers Source: *GEMS 1.18 Election Administrator's Guide, Revision 8.0*, unless otherwise indicated.	**Mandatory Client Security Policy Submitted to ITA** Source: *GEMS 1.18 TDP Appendix X: Client Security Policy*.
GEMS and GEMS client products operate on a stand-alone basis – and thus are not vulnerable to electronic intrusion – other than: • Downloading AccuVote-TS memory cards • Downloading AccuVote-OS memory cards • Downloading CTS vote centers • Uploading AccuVote-TS memory cards • Uploading AccuVote-OS memory cards • Uploading CTS vote centers • Running AccuVote-OS Central Count At no point are AccuVote-TS units, AccuVote-OS units, or CTS workstations connected to the internet in the course of voting or ballot counting. No wireless communication is enabled or employed between the GEMS server and any of the GEMS client devices. To restrict the possibility of unauthorized access to the GEMS server, modems connected to the server -through which polling location uploads will proceed should be powered on only in the course of upload testing, and following election close. Once all polls have uploaded, modems should be powered off. If the network used for downloading or uploading is physically integrated into a larger network, the components used for downloading should be isolated from the remaining network environment by means of a firewall. Any Election Reporting Client machines to which GEMS issues results (over an IP	The internal system addresses, configurations, and related system design information for networked computer systems must be restricted such that both systems and users outside the internal network cannot access this information. Any computers that can be reached by third-party networks (dial-up lines, value added networks, the Internet, etc.) must be protected by a privilege access control system approved by the client jurisdiction Information Security Department as well as Diebold Election Systems. This policy does not apply to computers which use modems to make outgoing dial-up calls, provided these systems do not receive unattended incoming dial-up calls. Public Internet servers must be placed on subnets separate from internal networks. Routers or firewalls must be employed to restrict traffic from the public servers to internal networks. All inbound dial-up lines connected to internal networks and/or computer systems containing computers should pass through an additional access control point (such as a firewall), which has been approved by the Information Security Department, before users reach a log-in banner. All in-bound real-time external connections to internal networks and/or multi-user computer systems containing servers, workstations, and other electronic devices must pass through an additional access control point (aka a firewall, gateway, or access server) before users can reach a log-in banner. No intranet servers, electronic bulletin boards, local area networks, modem connections must be established with computers without the specific approval of the client jurisdiction and Diebold Election

connection) should be secured within a firewall to prevent potential intrusion, either into the Election Reporting Client machines or the GEMS server. Since any FTP server to which an Election Reporting Client communicates will normally be resident outside of the firewall, unauthorized access to unofficial election results reports should be prevented by maintaining access to the FTP server secure. It will not be possible to access the GEMS database by means of an FTP server to which the Election Reporting Client communicates. The GEMS server is capable of transmitting data to the AccuVote-TS in encrypted format using Secure Sockets Layer (SSL) and Transport Layer Security (TLS) protocols. The purpose of encryption is to prevent the unauthorized interception and interruption of results uploads from the AccuVote-TS units to the GEMS server at election close. SSL/TLS transmission to the AccuVote-OS is not supported. AccuVote-TS memory cards should be programmed over a local area network connecting the GEMS server and AccuVote-TS units in a secure manner, isolated from the external network environment. At no time should memory cards downloaded with election information be left unattended, and should always be under the observation of at least two Chief election judges.	Systems. This policy helps ensure that all networked systems have the controls needed to prevent unauthorized access. Unattended active network ports which connect to the internal computer network must not be allowed in public areas including building lobbies, company cafeterias, and conference rooms readily available to outsiders. The election jurisdiction must maintain a current inventory of all connections between all election servers and workstations to external networks. Cable modems must not be used to connect to any servers and workstations unless a firewall and a virtual private network (VPN) is employed on the involved computers. Dial-up modems should not be connected to servers and workstations which are simultaneously connected to a local area network (LAN) or another internal communication network. No modem lines should be connected to computers or networks, unless these lines have first been approved by the client jurisdiction and Diebold Election Systems. Jurisdiction workers must not establish any communications systems which accept in-coming dial-up calls unless these systems have first been approved by the Director of Information Security. If a computer user attempting to gain access via a dial-up line has not provided a correct password after three (3) consecutive attempts, the connection must be immediately terminated.

***Table 6.7* Logs and Auditing Policy Discrepancies**

Logs and Auditing	
Diebold Security Policies Distributed to Customers Source: *GEMS 1.18 Election Administrator's Guide, Revision 8.0*, unless otherwise indicated.	**Mandatory Client Security Policy Submitted to ITA** Source: *GEMS 1.18 TDP Appendix X: Client Security Policy*.
All activities in GEMS are posted to the Audit Log. It is not possible to delete the Audit Log, or in any way alter it, other than by the automatic posting of an entry for every event to the log. Audit logs are also maintained in the same manner for the AVServer console, in which all AccuVote-TS, AccuVote-OS Precinct Count, and CTS transmission events are logged, Central Count Server console, in which all AccuVote-OS Central Count transmission events are logged Poster, in which all database posting-related events are logged Regional Server/Send Regional Results console, in which all regional transmission events are logged Every transaction on the AccuVote-TS unit is audited. Audit transactions are stored in chronological order, and may not be altered. Every memory card inserted into the AccuVote-TS unit is recorded in the Audit Log. Every memory card upload is recorded on the AVServer console in GEMS – the memory card entry is flagged as uploaded under the Vote Centers tab, while the event is logged under the Log tab. Failure of a transmission between an AccuVote-OS device and the GEMS server will be automatically logged in the GEMS AVServer console log Every transaction on the AccuVote-OS unit is audited. Audit transactions are stored in chronological order, and may not be altered. Every memory card inserted into the AccuVote-OS unit is recorded in the Audit Log. All equipment preparation, testing, and repair activities should be logged once they have been completed. Every programmed AccuVote-TS and	All user-ID creation, deletion, and privilege change activity performed by systems administrators and others with privileged user-IDs must be securely logged and reflected in periodic management reports. All production application systems which handle sensitive information must generate logs that show every addition, modification, and deletion to such sensitive information. Computer systems must securely log all significant security relevant events. Examples of security relevant events include: password guessing attempts, attempts to use privileges that have not been authorized, modifications to production application software, and modifications to system software. Logs of computer security relevant events must provide sufficient data to support comprehensive audits of the effectiveness of, and compliance with security measures. All privileged commands issued by computer system operators must be traceable to specific individuals via the use of comprehensive logs. All computer systems running production servers and workstations should include logs which record, at a minimum the following data: (1) user session activity including user-IDs, log-in date/time, log-out date/time, and applications invoked, (2) changes to critical application system files, (3) additions and changes to the privileges of users, and (4) system start-ups and shut-downs. Log-in passwords must not be recorded in system logs unless these same system logs encrypt the passwords. Computerized logs containing security relevant events must be retained for at

AccuVote-OS should be labeled. All administrative reports issued in the course of GEMS database development are reviewed, approved, and filed. Election results reports should be printed and filed once designated voting and Logic and Accuracy testing has been completed prior to election day. These reports confirm that the voting device counts and tallies votes as intended, and may be verified against manual tally sheets prepared for Logic and Accuracy testing process. The Audit Log is printed and filed once designated voting and Logic and Accuracy testing has been completed prior to election day. The act of setting memory cards to Election Mode by authorized election staff is logged by the same staff. Zero Totals reports printed at the outset of voting are signed by authorized election judges, attesting to the fact that all race and candidate counters are zero. Voters are logged in the pollbook as they are issued voter access cards. Election Totals reported printed at the conclusion of voting are signed by authorized election judges, attesting to the fact that the voting process proceeded correctly, and that results are present on the Election Totals report. All voting devices must be accounted for. All voter access card encoding devices must be accounted for. All smart cards and election supplies must be accounted for. All ballots must be accounted for. All Audit Logs are printed and archived following election close. PC-based Event Viewer logs should be reviewed and any anomalies accounted for. Voted memory cards are stored for the obligatory 22 months following the	least three (3) months. During this period, such logs must be secured such that they cannot be modified, and such that they can be read only by authorized persons. These logs are important for error correction, forensic auditing, security breach investigations, and related efforts. To assure that users are held accountable for their actions on production computer systems, one or more logs tracing security relevant activities to specific users must be securely maintained for a reasonable period of time. Application and/or database management system (DBMS) software must keep logs of user activities and statistics related to these activities which will allow them to spot and issue alarms reflecting suspicious business events. Mechanisms to detect and record significant computer security events must be resistant to attacks. These attacks include attempts to deactivate, modify, or delete the logging software and/or the logs themselves. All system and application logs must be maintained in a form that cannot readily be viewed by unauthorized persons. A person is unauthorized if he or she is not a member of the internal audit staff, systems security staff, systems management staff, or if he or she does not clearly have a need for such access to perform regular duties. Unauthorized users must obtain written permission from the Information Technology Manager prior to being granted such access. To allow proper remedial action, computer operations or information security staff should review records reflecting security relevant events on multi-user machines in a periodic and timely manner. Users of production servers and workstations should clearly informed which actions constitute security violations. Users must also be informed that such violations will be logged. A file naming convention must be

election. The GEMS database with final election results is backed up and archive Documenting the *chain of custody*, or flow of materials through the election lifecycle, should allow every significant event pertaining to a critical item in the election to be traced to a specific individual or set of individuals, place, and time. Chain-of-custody procedures pertain to the following critical election equipment: Election servers and clients AccuVote-TS units AccuVote-OS units Memory cards Voter Card Encoder units VCProgrammer units Supervisor cards Voter access cards AccuVote-OS ballots [The *Guide* presents samples of thorough chain of custody forms for use by election jurisdictions to track important events in the lifecycle of the election hardware and supplies.]	employed to clearly distinguish between those files used for production purposes and those files used for testing and/or training purposes. Every multi-user system should include sufficient automated tools to assist the security administrator in verifying the security status of the computer. These tools must include mechanisms for the correction of security problems.

Table 6.8 Vulnerability Testing Policy Discrepancies

Vulnerability Testing	
Diebold Security Policies Distributed to Customers Source: *GEMS 1.18 Election Administrator's Guide, Revision 8.0*, unless otherwise indicated.	**Mandatory Client Security Policy Submitted to ITA** Source: *GEMS 1.18 TDP Appendix X: Client Security Policy*.
No attempt is made to perform unauthorized activities with the voting devices. *(sic)*	Workers must not test, or attempt to compromise internal controls unless specifically approved in advance by the client jurisdiction and Diebold Election Systems. Users must not exploit vulnerabilities or deficiencies in information systems security to damage systems or information, to obtain resources beyond those they have been authorized to obtain, to take resources away from other users, or to gain access to other systems for which proper authorization has not been granted. All such vulnerabilities and deficiencies should be promptly reported to the Manager of Information Security.

Table 6.9 Data Modification Policy Discrepancies

Data Modification	
Diebold Security Policies Distributed to Customers Source: *GEMS 1.18 Election Administrator's Guide, Revision 8.0*, unless otherwise indicated.	**Mandatory Client Security Policy Submitted to ITA** Source: *GEMS 1.18 TDP Appendix X: Client Security Policy*.
GEMS database development should proceed according to authorized election development policy only.	Privileges must be established such that system users are not able to modify production data in an unrestricted manner. Users may only modify production data in pre\defined ways that preserve or enhance its integrity. In other words, users must be permitted to modify production data ONLY when employing a controlled process approved by management. Updates to production databases must only be made through established channels which have been approved by management.

Table 6.10 **Physical Security Policy Discrepancies**

Physical Security	
Diebold Security Policies Distributed to Customers Source: *GEMS 1.18 Election Administrator's Guide, Revision 8.0*, unless otherwise indicated.	**Mandatory Client Security Policy Submitted to ITA** Source: *GEMS 1.18 TDP Appendix X: Client Security Policy*.
AccuVote-OS firmware is locked inside the AccuVote-OS unit, and should only be accessible to authorized officials.	All clients and employees will receive this and all other related information security policies.
The memory card slot is provided with a cover plate. When this plate is in the closed position, it allows for a security seal to be placed in a hole in the post that the plate fits over. If there is any tampering with the memory card once it is behind this sealed cover plate, the tampering will be evident.	Advertising an area as secure should be minimized to decrease notification of a secure area to an unauthorized/malicious party. Structural protection should be maximized within the boundaries outlined within risk assessment.
Every compartment on a ballot box that is full and has been set aside in the course of voting remains locked, and a security plate locked into the ballot entry slot on the ballot box lid Limit access to voting machine keys.	Structural protection applies to all structures housing election equipment, such as a room where election equipment and servers are housed. Secure rooms should be constructed with full height walls. Secure rooms should be constructed with fireproof ceilings.
Access to voting equipment, in the warehouse or elsewhere, should be monitored and logged.	External access to secure rooms should be kept to the least number of privileged personnel.
Voting equipment should be stored according to designated storage conditions only. All security-related supplies, such as Supervisor cards, must be maintained secure at all times.	Secure rooms should contain a minimum number of solid, fireproof, and lockable doors required to maintain day-to-day operational efficiency. All points of entry/exit should be observable by authorized security staff.
Following the Logic and Accuracy Test, memory cards are locked and sealed into the AccuVote-TS unit, preventing any unrecorded, unauthorized access to the memory card. Access to ballot box and voting device keys is restricted to authorized pollworkers only.	All points of entry/exit in a secure room should have auto-closing devices and/or should never be left open. Any aperture within a point of entry/exit in a secure room such as a window should be reasonably small and provide for a locking mechanism.
The PCMCIA memory card representing primary election storage remains locked and sealed in the election data compartment in the course of voting.	A procedure should be put in place that allows for maintenance of appropriate locks.
Privacy panels on all AccuVote-TS	All points of entry/exit should remain locked when not in use in coordination with local fire policy. In the case of a breach, all locking mechanisms affected by the method of invasion should be

units are in an open position. Voting equipment is supervised by election judges in the course of election day. No one accesses voting booths other than voters that have been issued voter access cards. Full AccuVote-OS ballot boxes should be sealed. Voters do not leave the polling location with unauthorized materials. Voters do not leave the polling location with ballots or voter access cards. No unauthorized materials are present in or on the voting devices. The grounds of polling locations are maintained secure. No voter access cards are present in the voting area, other than those being held by valid voters. No unauthorized individuals are present at the voting location following election close. Once Voter Card Encoder units have been encoded, they should be stored in sealed, secure containers, and the seal numbers recorded. It is essential that all election facilities be maintained in a secure manner at all times, for the entire duration of the election equipment in the facilities, in the course of the election lifecycle, and beyond. Access to facilities containing election equipment should be limited to authorized election staff only. Every access to facilities containing election equipment should be documented. A sign-out sheet should be completed for every item of election equipment removed from or returned to the warehouse, including the device name, serial number, sign-out time, name and signature of the official signing out equipment, and the return time as well	changed. The method/device used to change the locking mechanism should effectively remedy the invasion tactic used by the unauthorized third party. Alternative physical security strategies should be periodically investigated and considered by the security staff. Where applicable, the use of secondary window restraint devices (e.g. window bars) should be considered. Where applicable, the use of anti-theft cabling with centralized alarm monitoring should be considered. Such scenarios would include, but not be limited to, any device containing, processing, or interacting with data/processes classified as PRIVATE or higher. The use of a more robust locking mechanism on all points of entry/exit should be considered. The system should include centralized monitoring of all door events and provide for two-factor authentication if possible (e.g. magnetic key cards, proximity cards, biometrics). The above said events provided by the system should be able to be categorized by the security staff and acted upon by a third party alarm/notification system. The use of a monitored alarm system in the secure room should be considered. The alarm system should provide for unauthorized entry notification and detected motion notification when activated. The service should be monitored by a third party security service if in-house security staff does not have the resources to monitor the system 24/7. The use of a monitored video surveillance system in the secure room should be considered to provide visual confirmation of an act of malice. The system should provide coverage of all points of entry/exit and all assets within the secure room classified as PRIVATE or higher. The coverage should be

as name and signature of official returning equipment. All facilities should be characterized by the following: Access limited to authorized individuals only Approved screening process for determining election official authorizations Proven tools and services to secure facilities, including: Physical door and window locks Electronic door and window locks, including: Password-based access Smart card-based access Biometric-based access Security guards Since voting devices, memory cards, AccuVote-OS ballots, and voter access card devices are normally warehoused separately from the election administration office, the destruction of the election administration office should affect the GEMS server, server communication devices, such as modems, and possible archive materials only.	configured in such a manner that a visual confirmation of an individual's profile and unmistakable facial/body characteristics can be attained. All video should be retained for the period of time needed to review by an authorized security personnel. The system should provide for a method to remove the video from the recording device onto another media for transportation or archival in the case of an incident being recorded. Proper disposal of confidential waste should be carried out in a careful and adequate manner to maintain confidentiality. A documented procedure should be distributed to all associates whose roles and responsibilities outline their handling of confidential material. This procedure should document in detail the use of a company provided facility that will properly and thoroughly destroy (e.g. using an industrial shredding device) the material. The documented procedure should also properly describe the destruction of materials based on their associated retention policy. Business critical systems should be separated from general systems. To prevent covert use, workstations not routinely used to display sensitive information should be stored in open, visible spaces. Maintain a secure inventory of all equipment and peripheral equipment, with up-to-date logs of manufacturers, models, and serial numbers. Consideration should be given to the use of videotape for insurance purposes. All mobile computing devices, such as laptops, should be locked in a secure cabinet while not in use. All computers should be logged off when the operator is not in the vicinity of the computer. Use a managed virus scanner on all

	computers at all times. A regular schedule for updates to the virus scanning engine should be documented and performed according to the virus scanner creator's virus profile release cycle. A regular schedule for full local and network virus scanning should be documented and performed. All third party computers that need to participate on your network should be inspected and virus scanned by a security staff before being connected. All peripheral devices should be distributed and implemented based on user's privilege level. A documented usage outline should be completed by authorized security personnel for every device used for day to day operations. A method should be devised to allow for centralized monitoring of problem detection on all devices. Equipment labeling should be created and implemented in covert and overt ways as to make unauthorized tampering more difficult.

Given the very short span of time for the Documentation Team's review of the TDP containing the Client Security Policy discussed above, it is not possible to conclude whether the security practices described in the Client Security Policy can be successfully and practically implemented in the use of Diebold voting systems.

6.2 Treatment of Security Issues in the Customer Documentation

The product documentation that Diebold distributes has significant information about the security features of its systems and, to a degree, how best to conduct election tasks in a way that leverages security features built in to the system. Election administrators can read the documentation to find out about some security hazards and prudent precautions. The documentation also includes useful sample forms for tracking the chain-of-custody of voting equipment and supplies. Some aspects of the documentation related to security, however, are cause for concern.

The documentation of security issues that is provided to the customers does not adequately explain the threats that the Diebold policies are designed to address. The coverage of security topics is generally limited to broad statements that certain threats are not realistic or that the recommended safeguards eliminate entirely the threats that they are designed to address. At many places in the customer documentation, the treatment of security issues amounts to categorical statements that the system is not vulnerable to a certain threat, and thus vigilance in that area is unnecessary. For example: "No system

functionality is accessible by connecting an external electronic device to an AccuVote-TS port, and no port is physically accessible when the Accu-Vote TSx is configured for voting on election day."[64] This comment suggests that no vigilance is warranted in the face of someone, perhaps a voter, connecting an external electronic device to a port on a voting machine. The comment also tends to suggest that the reader accept the inviolability of the rather flimsy lock and plastic door that guard the port compartments.

Similarly problematic are passages such as:

> GEMS and GEMS client products operate on a stand-alone basis – and thus are not vulnerable to electronic intrusion – other than:
>
> - Downloading AccuVote-TS memory cards
> - Downloading AccuVote-OS memory cards
> - Downloading CTS vote centers
> - Uploading AccuVote-TS memory cards
> - Uploading AccuVote-OS memory cards
> - Uploading CTS vote centers
> - Running AccuVote-OS Central Count[65]

Here, the documentation offers a brief list of possibly vulnerable operations that purports to be exhaustive. The ever expanding breadth of known techniques for electronic intrusion, the degree of interaction with the devices by members of the public and pollworkers, and the paucity of reliable detection capabilities renders this statement not only false but perhaps detrimental to achieving and safeguarding the system's security. It suggests that outside of these seven enumerated windows of vulnerability, no vigilance is needed.

In short, such reassurances do not seem to advance any security objective, but rather to impress the reader with the security of the voting device and reduce his vigilance. Similarly, comments such as "uploading a memory card provides no ability to alter the contents of the memory card,"[66] even if true, do not advance security so much as reassure the reader.

Persistently, the manuals do not explain the risks to be avoided by specific security policies. Thus, officials have no means of evaluating Diebold's proposed security policies. They cannot adapt policies to unique jurisdictional needs, applicable regulations, available resources, or otherwise make decisions about security policies independent from the vendor. The categorical dismissal of threats, coupled with a lack of information about the actual behavior of the voting systems in relevant contexts, both discourage independent assessments of security matters and deprive officials of the information required to undertake such assessments.

The potential harm caused by the promulgation of questionable or ill-suited security procedures is compounded by documentation statements including: "No Diebold Election Systems, Inc. software product should be used in a manner other than as intended, that is, in a manner contravening recommended usage procedures provided in Diebold Election Systems, Inc. product support documentation."[67]

---

64 *GEMS 1.18 Election Administrator's Guide, Revision 8.0*, p. 16-4. Statement is repeated in reference to the AccuVote-OS on page 16-7.

65 *GEMS 1.18 Election Administrator's Guide, Revision 8.0,* p. 16-13.

66 *GEMS 1.18 Election Administrator's Guide, Revision 8.0*, p. 16-5.

67 *GEMS 1.18 Election Administrator's Guide, Revision 8.0,* p. 16-12.

The customer documentation discussion of security issues also tends to mischaracterize or overstate the protection provided by certain safeguards. There are instances throughout the documentation in which tamper-evident seals are said to "prevent tampering."[68] Similarly, the use of encryption is said to prevent disruption of communications.[69] Though the voting devices' plastic doors with small barrel-key locks are easily bypassed, the doors are presented as effective in blocking access to the compartments inside. There are instances in which passages in the documentation are not merely unhelpful, but could potentially lead to overconfidence in the face of a potential threat, such as:

> Once the Logic and Accuracy Test is complete, memory cards are sealed into voting machines, a numbered seal is placed on the PCMCIA compartment door as well as the enclosing voting booth of all AccuVote-TS units, as well as over the memory card slot of all AccuVote-OS units, and the seal numbers recorded by election officials in order to prevent tampering of *(sic)* the units.[70]

6.3 Security Configuration of the GEMS Server Provided to the TTBR

As described in Section 4.3, we performed a configuration audit of the GEMS server that was provided to the TTBR red team testing room at the SOS facility in Sacramento. We were informed by Diebold technical personnel that the GEMS server was configured just as one would be when delivered to an election jurisdiction in California. Part of the configuration audit focused on the security settings on the TTBR server.

Among the security-related findings of our configuration audit was that none of the user account and password policies were active on the GEMS server. (See Section 6.1 for a discussion of password policies in the Diebold documentation.) Password policies such as minimum length, strength requirements, and history retention were not implemented.

We also found that none of the system-level auditing of security events was enabled, despite the fact that the GEMS configuration described in the *GEMS 1.18 Server Administrator's Guide* states that the security logging should be enabled and verified to be active. Our examination of other operating-system audit logs found that the Windows Application log, System log and Security log were all configured to retain records of events for only 7 days and that the logs were limited to 512 KB in size. Maintaining all logs of election audit information for a minimum time period (far greater than 7 days) is a requirement for election security audits and other examinations, as well as federal law. Care should be taken to make sure that California counties that have received GEMS servers configured by Diebold are not being placed in violation of federal or California law due to improperly configured logs on their GEMS servers.

It should be noted that, while it is advisable for election officials to configure security settings on systems in a manner consistent with approved security policies, the GEMS configuration documentation that is provided to customers does not describe the procedure for effecting the proper security settings. The customer documentation's sole mention of such settings is in the *GEMS 1.18 System Administrator's Guide*, which describes a process to view the logs[71] but none to configure them properly. An industrious election official may undertake to fix relevant settings upon viewing that the settings were not correct, but

---

[68] For examples, see *GEMS 1.18 Election Administrator's Guide, Revision 8.0*, p. 2-6 and p. 4-133.

[69] *GEMS 1.18 Election Administrator's Guide, Revision 8.0*, p. 16-12.

[70] *GEMS 1.18 Election Administrator's Guide, Revision 8.0*, p. 2-6.

[71] The document directs the reader to the Windows Event Viewer and instructs, "[c]lick on the Security entry in the left-hand display panel, and observe the system related events that appear in reverse chronological order in the right-hand display panel." *GEMS 1.18 System Administrator's Guide, Revision 6.0*, p. 3-8.

would do so with no help from the Diebold documentation. The only document that offers any significant coverage of platform security configuration is the *GEMS 1.18 Server Administrator's Guide*, which is not to be circulated outside of Diebold.[72]

The following two tables present the system security related findings of our configuration audit of the GEMS server provided to the TTBR study.

Table 6.11 **Selected Security Settings on the TTBR GEMS Server**

Selected Security Settings	
Policy	**Effective Setting**
Audit access of global system objects	Disabled
Audit use of backup and restore privilege	Enabled
Allow system to be shut down without having to log on	Disabled
Automatically log off users when logon time expires	Enabled
Digitally sign client communication	When possible
Digitally sign server communication	Disabled
Disable CTRL+ALT+DEL requirement for logon	Enabled
Do not display last user name in logon screen	Enabled
Prevent system maintenance of computer account password	Disabled
Recovery console: allow automatic administrative logon	Disabled
Recovery console: allow floppy access to all drives and all folders	Disabled

Table 6.12 **Password Policies on the TTBR GEMS Server**

Passwords	
Enforce password history	0 passwords remembered
Maximum password age	42 days
Prompt user to change password before expiration	14 days
Minimum password age	0 days
Minimum password length	0 characters
Passwords must meet complexity requirements	Disabled
Account lockout duration	Not defined
Account lockout threshold	0 invalid logon attempts
Reset account lockout counter after	Not defined

[72] "The *GEMS 1.18 Server Administration Guide* is intended for technical support staff internal to Diebold Election Systems, Inc. only. It is not intended for use external to Diebold Election Systems, Inc. The document provides a detailed framework for the configuration of GEMS servers." *GEMS 1.18 Server Administration Guide, Revision 3.0,* p. 1-1.

Table 6.13 Event Logging Settings on the TTBR GEMS Server

Event Logging	
Audit account logon events	No auditing
Audit account management	No auditing
Audit directory service access	No auditing
Audit logon events	No auditing
Audit object access	No auditing
Audit policy change	No auditing
Audit privilege use	No auditing
Audit process tracking	No auditing
Audit system events	No auditing
Application Log - Maximum log size	512 KB
Application Log - Overwrite events older than	7 days
Security Log - Maximum log size	512 KB
Security Log - Overwrite events older than	7 days
System Log - Maximum log size	512 KB
System Log - Overwrite events older than	7 days

Table 6.14 Possibly Unneeded Services on the TTBR GEMS Server

Possibly Unneeded Services	
Service	**Start Setting**
NetMeeting Remote Desktop	Manual
Remote Registry Service	Manual
Telephony	Manual (was running at the time of configuration audit)
Telnet	Manual
Windows Installer	Manual
Wireless Configuration	Manual

Conclusion: Vendor Nonconformity Responses

The Diebold documentation is incomplete and faulty in certain ways. The Source Code and Red Teams have identified some major vulnerabilities with key components of the Diebold VS.

In addressing Diebold voting systems' areas of potential nonconformity with applicable standards, regulations and conditions of certification, election officials may wish to be aware of the Diebold corporate policy to nonconforming products. In case remediation is sought, the following policy might be useful:

From the **Diebold North America: Quality Systems Manual** QSM00001 Version 15.0 ["QSM"]

In the "Control of Nonconforming Product" material, the QSM states that "Diebold ensures" that any "product which does not conform to product requirements" will be "identified and controlled to prevent its unintended use or delivery." The manual directs further consultation for "controls and related responsibilities and authorities" to QSP00013, Control of Nonconforming Product.

The QSM represents that "nature of nonconformities and any subsequent actions taken, including concessions obtained" are the subject of company records. It further warrants that if a "nonconforming product is detected after delivery or use has started," Diebold will undertake "action appropriate to the effects, or potential effects" of the product's failure to confirm.

Other Diebold North America Manuals relevant to dealing with nonconforming products include QSI00025, Global Manufacturing Customer Relations Process and QSI00002, Field Change Order (FCO) Process.

If the California SOS identifies any material deficiencies or departures from the operative product requirements, these manuals' policies may prove relevant. [73]

---

[73] Section 6.1 of this Documentation Report contrasts the mandatory client security policy ("Appendix X") that was part of the vendor's confidential submission to CIBER for Federal ITA qualification testing with the security policies set forth in the documentation provided to customers, pointing out areas in which they significantly differ or conflict. Appendix X mandates election jurisdiction security policies and practices that are significantly stricter.

On August 24, 2007, the Cal-SOS informed us that it had just learned that, in possible contrast to the vendor documentation this team was provided for the TTBR evaluation, the vendor has stated that it has made available to California counties a Client Security Policy (CSP) document. The vendor (formerly Diebold, now Premier Election Solutions, Inc.) stated that beginning in 2004 as part of a legal settlement, it has provided, upon request, a CSP document to California counties that have purchased its election systems. The vendor further stated that a CSP document has been included on product documentation CDs beginning sometime in 2006.

The CSP that the vendor states has been disseminated to California counties is not referenced in any of the documents that were provided for the TTBR, nor was the team provided a copy of this CSP document to review. We are thus unable to determine whether the CSP reflects precisely the mandatory security policies that the vendor submitted for its system's certification and qualification reviews, or compare it to the vendor's other security policy documentation. *[Added to the original Report on August 27, 2007]*

DIEBOLD

Diebold Election Systems, Inc.
P.O. Box 1019
Allen, Texas 75013
469-675-8990
fax 214-383-1596
e-mail www.dieboldes.com

Via Certified Mail
RRR 7006 0810 0004 3838 5559

August 22, 2007

The Honorable Debra Bowen
California Secretary of State
1500 11th Street, 6th Floor
Sacramento, CA

Re: DESI/Premier Response to Red Team Review

Dear Honorable Secretary Bowen:

First, as you have probably heard, Diebold Election Systems, Inc. is changing its name to Premier Election Solutions, Inc. This should be effective within a week or so and we will be filing relevant documents with other parts of the Secretary of State's organization shortly thereafter, probably by way of our agent, CT Corporation.

Please find enclosed the responses of Diebold Election Systems, Inc./Premier Election Solutions ("Premier") to the California Red Team review document released July 27, 2007, in respect to the use of "GEMS 1.18.24/AccuVote-TSX/AccuVote-OS/DRE & Optical Scan Voting System."

The final document with Premier's response was made available today to our customer jurisdictions in California. If you or your staff has any questions regarding Premier's response, please contact me at your convenience.

Sincerely:

Dave Byrd
President
Diebold Election Systems, Inc.

Cc: Kathy Rogers (DESI)
Michael Lindroos (DESI)
Don Vopalensky (DESI)

Report of Diebold Election Systems, Inc. ("DESI") to

California Secretary of State Red Team Report Issued on the GEMS 1.18.24/AccuVote-TSX/AccuVote-OS/DRE & Optical Scan Voting System on July 27, 2007.

DESI has announced it is in the process of a name change to Premier Election Solutions, Inc. Accordingly, the company name referenced hereafter is "Premier".

Issued on this date of 8-20-2007

Background

In March 2007, following through on a commitment she made while campaigning for the office, Secretary of State Debra Bowen announced her intention to conduct a full "Top to Bottom Review" of all voting systems currently in use in California. In May of 2007 Secretary Bowen entered into an interagency agreement with the University of California to conduct the review. Of the eight voting systems originally selected for the Top to Bottom Review, only three were ultimately tested under this plan.

As required by the state, prior to beginning the test, Premier provided all applicable systems, full documentation, complete source code, and all passcodes required to gain full, unfettered access to the system. In addition to providing all of the above, experts from Premier also spent several hours explaining the system in great detail to the Red Team members, information which allowed them to more knowledgably carry out their attacks.

This document contains responses to the Red Team report which was issued by eight computer scientists at the University of California who spent six weeks with full access to the system in an exercise that included no physical or operational security controls.

1. GEMS Server Vulnerabilities

The report prepared by the Red Team identifies several GEMS server vulnerabilities, as addressed below.

a. Windows Vulnerabilities

The Red Team review noted what it described as "stark" discrepancies between the setup of the GEMS server provided for evaluation and the correct GEMS server configuration as described in the Premier documentation [ref: *GEMS Server Configuration Guide*] that was provided, and then proceeded to describe various known security vulnerabilities in the Windows 2000 operating system on the incorrectly-configured server. This discrepancy existed because the SOS required the servers to be configured the same as certified (approximately two years ago) for use in California, whereas the documentation contains the Windows security patches and additional security configurations Premier has defined since certification of the system by the state of California. Since 'any modification to the system' must be certified, Premier was not allowed to install the patches or change the security settings on the system.

Had the Red Team configured the server as per the supplied documentation, the Windows vulnerabilities associated with an un-patched server that were identified in the report would of course be mitigated. Furthermore, standard Windows logging features would have been correctly enabled, as indicated in the configuration guide.

The Red Team also claims to have "uncovered evidence" that Premier technicians created a particular remotely-accessible Windows account, and that there is further "evidence" to suggest that this account is intended to be used by TSx units for dial-in access at the close of polls on Election Day. TSx uploads are in fact performed via a Remote Access Service (RAS) connection, as documented in section 4 of the *GEMS 1.18 System Administrator's Guide*, and the presence of a dedicated account for this purpose should not be construed as either surprising or covert. Instructions for configuring and/or disabling RAS, including specification of RAS-enabled accounts, are documented in the System Administrator's Guide. The possibility of system exploits via the RAS service should be mitigated by applying the policies defined in the *GEMS Server Configuration Guide*, and by evaluation and application of appropriate operating system security updates as they are made available by Microsoft.

b. GEMS Databases

The Red Team used Windows Administrator access to delete, corrupt, and modify GEMS databases. It is obvious even to a layperson that a user with full administrative access to any machine, whether locally or via a network connection, will of course be able to delete or intentionally corrupt virtually *any* file on that system. This is as true in other operating systems as it is in Windows.

As noted in the product documentation, the version of GEMS submitted for use in California requires restrictions on access to the GEMS server and/or other procedural steps (e.g. creating a secure backup of the GEMS database on write-once media such as a CD-ROM) to protect the GEMS database against unauthorized modifications [ref: *Client Security Policy*]. As noted in the Red Team report, the review was conducted under assumptions of full access to the system and the absence of all procedural safeguards and therefore the reviewers were able to modify the database in an unauthorized manner. The specific issue concerning being able to read and/or modify the contents of the GEMS database file directly has already been addressed in a newer revision of the GEMS software that was not reviewed by the Red Team for use in California.

c. GEMS Audit Logs

The Red Team notes that it found methods for executing actions from within GEMS that could not be tracked via the GEMS audit logs. Upon review of the issue Premier agrees with this statement and will address the logging problem identified in the report. The report seems to imply, however, that these actions may be executed via the GEMS application leaving no trace whatsoever in the audit log. It should more accurately be

noted that what is *actually* left in the log as a result of this attack is a series of extremely suspicious entries whose entry text is obscured, and which would certainly invite further investigation.

d. GEMS Election Configurations

The Red Team correctly identified a class of format string vulnerability that, when exploited, could cause a TSx unit to function incorrectly under certain identified circumstances. This problem is caused by an implementation error in the TSx software and will be corrected in the next release. It should of course be noted that this type of election data problem is exactly the sort of issue that standard logic and accuracy testing is intended to detect. As noted earlier, the Red Team conducted its review in the absence of any standard procedures and was consequently able to exploit this particular bug without the problem being detected before the simulated Election Day.

2. GEMS Server Networking Components

This section of the review appears to be poorly-worded: it describes how *a priori* knowledge of the Windows Administrator credentials permitted the reviewers to guess the authentication credentials for a particular piece of networking hardware. Guesswork is of course entirely unnecessary; the credentials for the networking hardware in question are provided in section 5 of the *GEMS 1.18 System Administrator's Guide*. The report then implies that the discovery of those credentials would provide sufficient system access to "manipulate every setting on the networking devices and on the server," and further notes that the Red Team was subsequently able to use this access on the host system to install the drivers for a USB wireless dongle. This entire discussion is not at all accurate: discovery of the authentication credentials on the network device in question would *not* provide any ability to manipulate any settings on the host server at all. The ability to manipulate every setting on the host server through the Windows Administrator credentials was actually *provided to the Red Team at the outset of their review*; as noted in the introduction to the report, the review was predicated on (among other things) the assumption of full physical and administrative access to the GEMS server. As noted earlier, a user with full administrative access to *any* machine will of course be able to modify any settings on that system. This is as true of any other operating systems as it is in Windows. The ability to manipulate settings of the networking device, and the default credentials required to do so, are documented in the *GEMS 1.18 System Administrator's Guide* as described above.

As noted in the product documentation, the system requires restrictions on access to the GEMS server and/or procedural safeguards to protect access to the system via exposed ports on the machine [ref: *Client Security Policy*]. As noted in the Red Team

report, the review was conducted under assumptions of full access to the system in the absence of all procedural safeguards and therefore the reviewers were able to install the drivers for a USB wireless dongle, much as they would be able to install anything else they deemed appropriate.

3. Precinct Count AV-OS

The Red Team notes that it was able to verify the findings of some previous reports. This is hardly surprising; the state of California had already made such a determination in a previous review of the same software and was (re-)apprised of the specific issues by Premier in advance of the Red Team review. The specific issues identified in this area have already been addressed in a newer revision of the AVOS firmware that was not reviewed for use in California.

The Red Team further notes unspecified "denial of service" types of attacks that, if executed, would render an AV-OS unit unusable for the remainder of Election Day. It seems obvious that most voting equipment is not indestructible and that rendering it inoperable by physically damaging it on Election Day is certainly within the realm of the possible. This sort of attack is not even unprecedented: in previous elections, voting machines have been damaged and rendered unusable by various types of physical assault. Counties of course have procedures in place to handle the case of machine failure, as typically a replacement machine is quickly deployed to the polling place in question. As noted previously, the Red Team review was conducted in the absence of any procedural considerations and under that premise any type of machine failure will of course terminate any given scenario.

4. TSx

The report prepared by the Red Team describes a number of issues with the TSx voting terminal. These issues are listed below.

a. TSx: Physical Security

The Red Team notes that it was able to violate the physical security of every aspect of the TSx unit. Premier-recommended procedures specify the use of tamper-evident seals in order to detect this sort of unauthorized access. As noted previously, the Red Team review was conducted in the absence of any procedural safeguards.

b. TSx: Malware

The Red Team notes that it was able to verify the findings of some previous reports. This is hardly surprising; the state of California had already made such a determination in a previous review of the same software and was (re-)apprised of specific issues by Premier in advance of the Red Team review. The specific issues identified have already been addressed in newer revisions of the TSx bootloader and application software that were not reviewed for use in California.

The Red Team theorizes a scenario by which malware could be virally propagated from a TSx unit to a GEMS server via format string errors in the result upload protocol. Premier rejects this particular claim and notes that no viable exploit of this type is actually described anywhere in the report.

c. TSx: Escalation of Privileges

The Red Team correctly identified an incorrectly-implemented error handling routine that can be exploited to escalate the privileges of the attacker to those of poll worker or central administrator. Premier acknowledges the issue and has already corrected the problem in a revision of the software that was not evaluated by the Red Team for use in California. The Red Team review actually reports many (at least twelve) variations of this specific issue. Premier certainly acknowledges that unauthorized escalation of the user's privilege level to that of poll worker or central administrator opens the door to assorted unauthorized actions by the attacker. Given that all such reported variants of privilege escalation are associated with a single exploit, however, itemizing a long list of variants on the theme, as in the Red Team report, seems gratuitous.

d. TSx: Default Static Key

As noted in the Red Team report, Premier documentation strongly urges election officials to generate their own election-specific keys and provides a visual indicator to alert those officials if they have not. The Red Team report takes issue with the visual indicator. Premier recognizes the argument made by the reviewers but respectfully disagrees with their conclusion on this point.

e. TSx: Malicious Voter Input

The Red Team report indicates that multiple voter-accessible input fields on the TSx are susceptible to malicious input and makes claims of observing erratic behavior with proof-of-concept code. Premier would note only that the full report lists no such attack, and that the only voter-accessible input field aside from the checkbox next to each ballot selection is the write-in candidate text. Vague claims of exploits in multiple but unspecified voter-accessible input fields are consequently difficult to address.

f. TSx: VVPAT

The Red Team review describes an attack scenario in which a "common household substance" could be used to deface, obscure, or destroy the VVPAT paper ballot records. What the public report did not state was that the lock on the VVPAT needed to be picked and the pickup canister needed to be pried open to facilitate this attack. Premier acknowledges that a wide variety of household substances will indeed deface, obscure, or destroy paper ballots. In the case of the TSx, those ballot records can be retrieved via the dual electronic copies of each ballot record. Obviously in cases where the paper ballot is the *only* record of the voter's intent, defacing of the ballots is an unrecoverable scenario. This same attack vector was inexplicably absent from the assorted Red Team reviews of paper-only voting systems.

The Red Team review also describes and attack scenario "that can put the Voter Verified Paper Trail (VVPAT) out of service until the unit is rebooted." This attack required the attacker to somehow turn off and back on the system (reboot it) while in the voting booth.

g. TSx: PCMCIA card

The Red Team notes that it was able to verify the findings of some previous reports. Specific issues were not identified but the previously published known issues have been reviewed and addressed where appropriate in a newer version of the software not reviewed for use in California.

Successful Attack Scenarios

Attack Scenario 1

Even ignoring the extreme logistical difficulties involved in exploiting the reported escalation of privilege exploit undetected in full view of the entire polling place, the premise of the attack is flawed in its own right.

a) "Deleting" election results via administrative access as described in the Red Team review does not actually delete the results, which can be retrieved from either the PCMCIA card or the voting machine's internal storage if necessary (as in this scenario). The Red Team's implication that the election results can be permanently erased via this mechanism is simply incorrect and most likely stems from an incorrect understanding of the feature.
b) Restarting the election on the unit will cause a new zero totals report to be printed. One can imagine that even the most lax pollworker is likely to notice a lengthy report being printed on a voting machine attended by a purported voter in the middle of Election Day.
c) Unlike the Red Team's unrealistic test environment, procedural safeguards do apply in the real world and cannot simply be ignored when it's convenient to do so. Archiving the current election results and resetting the election will also reset the public count on the machine, and pollworkers are instructed to periodically check the public count on each voting machine for various reasons. It is disingenuous for the reviewers to describe an attack conducted in a procedure-free test environment as though it were an attack that was successfully executed in a real-world environment, as they do here.
d) Chemical warfare attacks against the VVPAT printer involve either picking the lock on the printer door or removing the printer cover by force, lifting the cover and introducing the chemicals into the unit's security canister without breaking any tamper-evident seals or drawing attention to the attacker, all while in full view of the polling place personnel. No additional comment seems necessary.
e) Destroying the paper records in the VVPAT canister can be mitigated by re-printing the records from the electronic data. Obviously in jurisdictions that have chosen to simply assume the electronic records are not to be trusted, this option may not be possible. In most jurisdictions, however, recovery via this mechanism is an available mitigation option.

Attack Scenario 2

This is simply a variation of attack scenario 1 and the same comments (absent issues specific to chemical-based attacks on the paper ballot records) apply. The reviewers

misunderstand the effect of tagging data for deletion via the unit's administrative options and mistakenly assume that the data is unrecoverable. Moreover:

a) Printing extra ballot records is an extremely noticeable operation, given the speed of the printer and the amount of noise made by the VVPAT security canister during its operation.
b) Recovering the election data provides a correct indication of the correct number of ballot records to be recovered from the tape. Those ballots will be the first set of ballots on the tape; the duplicates are bounded at the top by the last valid ballot and at the bottom by the new zero report printed when the election is reset. Consequently the fraudulent ballot records can be detected and removed.

As described above, the real world necessarily reflects procedures that handle various scenarios that may arise, unlike the Red Team test environment which omits all such considerations for the sake of convenience.

Attack Scenario 3

This describes a potential ballot box stuffing attack via the same escalation of privilege exploit in the previous two attack scenarios. Ballot box stuffing has a well-documented history in paper-only elections and is obviously much simpler to execute in that environment. As stated previously, Premier has acknowledged this escalation of privilege issue and has already addressed the associated exploit in a newer software release, which will effectively eliminate all of these assorted variants on this theme. This new release of software was not evaluated for use in California. In addition to completely mitigating the previous two attack scenarios, resolving this exploit also fully addresses attack scenario 3.

Attack Scenario 4

This attack is a denial-of-service attack based on a physical assault on the AVOS scanner hardware. In the Red Team test environment such an attack has the implications described in the review, i.e. it "does remove the AV-OS from service for the remainder of Election Day." In a more real-world scenario the county would certainly have an equally low-tech mitigation strategy for such a low-tech attack: typically a replacement unit can simply be deployed to the polling place rather than having to resign oneself to having a non-functional unit for the rest of the day.

Potential Attack Scenarios

While these potential attack scenarios are rooted firmly in speculation by the report authors, it would be remiss not to comment on them.

Potential Attack Scenario 1

This is simply another variation on the same escalation of privilege exploit used in attack scenario 1, 2 and 3 above. As described above, the resolution of that specific issue in the next software release eliminates the TSx component of this potential attack as well as all other variants on the theme.

The second element to this potential attack is predicated on an incorrectly-configured GEMS server as described in the section on GEMS Server Vulnerabilities. Configuring the server correctly, as described in the server configuration guide supplied by Premier [ref: *GEMS Server Configuration Guide*], as well as instituting a regimen of regularly evaluating and applying operating system security updates when they are made available by Microsoft, will mitigate remote-access based escalation of privilege attacks on the GEMS server itself. Disconnecting the GEMS server from the phone system outside of election night, as recommended, will also limit attacks based on remote access.

Potential Attack Scenario 2

The specific issues identified have already been addressed in newer revisions of the TSx bootloader and firmware that were not reviewed for use in California. Moreover, Premier rejects the theorized virus propagation from the TSx to a GEMS server via upload of election results; there is no credible mechanism described for such an attack and no actual exploit is supplied. While Premier evaluates all claims made about its system and acknowledges actual issues that are brought to our attention, we must by the same token reject claims that simply have no basis in fact or are based purely on speculation.

Conclusion

Simply stated, it is a known fact that **any** voting system can and will be found vulnerable to an attack if the system is left open to full access. This is true of paper, punch card, optical scan, lever and electronic voting. There is no voting system in existence, nor is it possible to envision one, that could withstand unfettered and unmonitored attacks in a laboratory setting whereby dozens of computer experts spend hundreds of hours, with full access, attempting to infiltrate the system. This was confirmed by Matt Bishop, California "red team" leader:

"Our task was to analyze the machines, but those machines are just one piece of what makes an election secure," said Matt Bishop, professor of computer science at UC Davis, who led the (California) Red Team review. "In my 30 years in this field, I've never seen a system that was perfectly secure, but proper policies and procedures can substantially improve the security of systems." ScienceDaily.com July 31, 2007

This type of testing called "Red Team" is typically counterbalanced with a "Blue Team" which serves to provide the human intervention factor that exists in a real world election environment. A far more realistic test would have applied California elections laws and documented security and use procedures to the findings of the Red Team.

To review, it is important to note that many of the responses above confirm that numerous reported issues have been corrected in newer versions of software. Under the rules established for the Top to Bottom Review these newer versions of software were not allowed to be reviewed for testing.

7.1050

Software Review and Security Analysis of the Diebold Voting Machine Software

Ryan Gardner Alec Yasinsac Matt Bishop Tadayoshi Kohno
Zachary Hartley John Kerski David Gainey
Ryan Walega Evan Hollander Michael Gerke

Security and Assurance in Information Technology Laboratory
Florida State University
Tallahassee, Florida

July 27, 2007

Final Report
For the Florida Department of State

Software Review and Security Analysis of the Diebold Voting Machine Software

Table of Contents

Software Review and Security Analysis of the Diebold Voting Machine Software Final Report

1 Executive Summary

On May 14th 2007, the Florida Department of State (FLDoS) commissioned an independent expert review of Diebold Voting System Software, as documented in their Statement of Work [1]. The team, led by Florida State University's (FSU) Security and Assurance in Information Technology (SAIT) Laboratory, was commissioned to conduct a software code review as part of the state's voting system certification process. This report is the culmination of that review.

1.1 The Analysis' Scope

The scope of the investigation, as defined in the Statement of Work (SoW), is:

> *This review is for the purpose of yielding technological data to DOS [FLDoS] to ensure voting system effectiveness and security in Florida elections by investigating for potential flaws in target software as documented in reported literature and other published studies [1].*

The team constructed a flaw list from surveyed literature and this list drove the analysis. FLDoS provided the team fully functional hardware and accessories, which we utilized to test and confirm the code's operation. We did not conduct a comprehensive software review nor penetration testing, as each of these was outside the project scope.

We emphasize that our technical analysis reflects neither endorsement of, nor opposition to, certification. We present this technical data for consideration by FLDoS in their decision processes.

1.2 Systems Analyzed

The two primary systems analyzed consist of the Diebold Optical Scan, firmware version 1.96.8, and Touch Screen, firmware version 4.6.5. We also examined the Diebold Touch Screen bootloader version 1.3.6 as well as GEMS server software version 1.18.25. We considered flaws in previous versions of the software for all parts of each system, including those found in the AccuBasic interpreters.

1.3 Findings Summary

Our primary findings are:

The version of the Optical Scan and Touch Screen software that we examined:

(a) fixed many of the flaws in earlier versions, but

(b) retain significant flaws that are documented in this report.

As an example of the issues that remain, flaws in the Optical Scan software enable a type of vote manipulation if an adversary can introduce an unofficial memory card into an active terminal before the voting (or early voting) period (e.g., during "sleepover"). Such a card can be preprogrammed to alter the correspondence between physical bubbles on the scanned paper ballots and the candidates with which they are associated. Specifically, it can be used to essentially swap the electronically tabulated votes for two candidates, reroute all of a candidate's to a different candidate, or tabulate votes for several candidates of choice toward another chosen candidate. We implemented this attack in the laboratory. The attack succeeds despite new protection mechanisms apparently designed to protect against similarly-documented attacks in previous studies.

Many reported flaws were removed from the Touch Screen software. Nonetheless, we identified many that still exist. As one example, we found an attack that allows an adversary to prepare official, activated voter smart cards that would enable voters to cast multiple ballots in a ballot-stuffing attack. Creation of the cards requires an adversary able to insert a custom smart card into a legitimate voting terminal and to read the data off of a valid voter card (these steps could be done by separate

adversaries.) Once the adversary obtained the necessary information in this way, she could then create smart cards that could be used at any precinct throughout a county. Even if detected, this attack is not correctable: the malicious ballots, either in electronic or paper form, are essentially unidentifiable and thus cannot be removed.

We provide our detailed findings and supporting analysis in the sections below.

2 Project Introduction and Background

2.1 Report Organization

This document is the project report. It contains all of our pertinent findings and conclusions and the technical analysis that supports these conclusions. The document is written in two parts. The public part (Sections 1-7 and Appendix A) constitutes the public report in its entirety; it contains our findings and the analysis to support these findings. It is intended for public dissemination. In accordance with the terms of the Statement of Work, we have avoided revealing proprietary information in the public part of the report, and we are careful to avoid revealing information that would describe how to attack an election. The public report stands on its own and reflects the totality of our findings regarding known flaws.

The private part consists of Appendix B and Appendix C, which are confidential as required by the Statement of Work because (a) they contain vendor-proprietary information; (b) they contain information about flaws that are not relevant to this investigation; or (c) they describe how to mount security attacks in detail and thus are not appropriate for the public report.

This document first gives background information about the known flaws that guided our analysis. We then describe our findings and conclusions.

2.2 Terminology

2.2.1 SAIT Team. This phrase refers to the investigators organized for this project by SAIT Laboratory. The team members are identified in Section 2.3 below.

2.2.2 Flaw/Fault/Vulnerability/Threat. In this report's context, these four terms are closely related, distinguished only by subtleties and nuance. A software *flaw* is a defect that may or may not have an impact on normal program operation. We use the terms *flaw* and *fault* interchangeably. Software *vulnerability* is a code state or status that may lead to improper program operation. Thus, a flaw may introduce a vulnerability. The term *threat* refers to a potential occurrence that exploits a vulnerability, and is sometimes loosely used to refer to the threat manifestation.

2.2.3 Adversary/Attacker/Intruder. We use these terms interchangeably to refer to any malicious party that may try to manipulate a voting system. Our analysis focuses on two attacker categories, based on their role in the voting process. The first prospective attacker class is *Voter*. Three important voter properties are: (1) there are many voters; (2) they are untrusted; and (3) they have very limited access to the system.

The second major prospective attacker category is *Poll Worker*. There are fewer poll workers than voters, but they have greater system access in terms of time and capability.

Elections officials and voting system vendors represent two other recognized potential attacker categories. However, we consider attacks by these parties to be largely outside the scope of this review.

2.2.4 Adversarial Skill. In this report we shall often refer to resources required by an adversary, one of which might be technical sophistication. For example, in Section 3.6 we state that an adversary should have the requisite knowledge and skills. When we make such statements, we generally speak of the *first* adversary to develop the attack, *not* necessarily the individuals carrying out the attack. In many cases, the individuals carrying out the attacks do not need to have the same knowledge and skills

as the individuals developing the attacks.

2.2.5 Fixed. A flaw is *fixed* if the corresponding vulnerability is no longer present in the source code. We emphasize that it is impossible to identify all flaws, so any statement in this report that an item is fixed simply reflects our best professional opinions.

2.2.6 Benign. A flaw is *benign* if its technical effect does not interfere with the intended operation. Note that this is our technical assessment based on the extent of this study and the assumptions stated herein. Readers should consider all reported and potential flaws with a broader eye for system impact, in particular taking into account the specific policies and procedures under which the systems will be used.

2.2.7 ".abo" File. The Diebold AccuBasic Object scripts reside in files whose extension is .abo. We occasionally refer to AccuBasic Object script programs as .abo files.

2.2.8 TSX™/OS™/AccuVote. This report uses numerous voting machine model identifiers. We generally refer to the Diebold Touch Screen system as the TSX and the Optical Scan device as the OS. Both are members of the Diebold AccuVote series.

2.3 The Software Review Team

2.3.1 Senior Investigators

Matt Bishop is a Professor of Computer Science at the University of California at Davis. He is an expert in secure software and electronic voting systems, having participated in several widely recognized electronic voting software systems code reviews. His computer security textbook, *Computer Security: Art and Science*, is the acknowledged benchmark against which all others related to this topic are measured.

Tadayoshi Kohno is an Assistant Professor of Computer Science and Engineering at the University of Washington. He is an expert in cryptography and secure software and is an author on the seminal "Hopkins Paper" [8] that triggered the current movement to more aggressive voting system code review.

Alec Yasinsac is an Associate Professor of Computer Science at Florida State University, a co-Director of SAIT Laboratory, and is the lead Principal Investigator on this project.

2.3.2 Investigators

David Gainey is a computer science graduate student, a member of SAIT Laboratory, and is a member of the technical staffat the Florida State University Office of Technology Integration.

Ryan Gardner is a doctoral student at Johns Hopkins University and a member of the ACCURATE center for voting systems research.

Michael Gerke is a computer science graduate student and a member of SAIT Laboratory at Florida State University. He is presently employed by the Florida Department of State, Division of Elections.

Zachary Hartley graduated from FSU with a Computer Science Masters degree in April 2007. He is presently employed by the Florida Department of State, Division of Elections.

Evan Hollander is a computer science graduate student, a systems administrator in the Computer Science Department, and a member of SAIT Laboratory at Florida State University.

John Kerski is a computer science graduate student and a member of SAIT Laboratory at Florida State University.

Ryan Walega is a computer science graduate student and a member of SAIT Laboratory at Florida State University.

2.3.3 Team Organization

2.3.3.1 Internal Team Structure and Operation. All investigators conducted hands-on source code analysis. The list of previously noted flaws drove the analysis. Members received analysis assignments and were also free to investigate independently. Several investigators exercised automated analysis tools. The Lead PI coordinated analysis activities.

Team members conducted structured note taking and referred to these notes during final report preparation.

2.3.4 External Communication and Coordination

2.3.4.1 Florida Department of State (FLDoS). As noted in the SoW, FLDoS was entitled to observe the code review process at their discretion; they chose to limit their interaction. FLDoS only interacted with the team at our invitation. They proved to be a valuable information resource, providing election configuration files, general election knowledge, and hardware demonstrations to support our analysis. Their support was consistently prompt and complete. The FLDoS placed no restrictions on our activities within the SoW.

2.3.4.2 Diebold Election Systems. The SAIT Team established a communication channel with the vendor within the first project week and interacted with them on several occasions. The purpose of these communications was for the SAIT Team to gather information and to clarify issues relative to the target systems and source code.

The relationship with Diebold Election Systems was professional and cordial. Diebold Election Systems assisted the SAIT Team by providing specific feedback relative to the gathered flaw list and also provided the team a copy of the private appendix to the 2006 California VSTAAB review [2]. We also conducted a conference call between the SAIT Team and Diebold engineers that facilitated our review.

2.3.4.3 Florida State University. FSU and SAIT Laboratory hosted the code review and provided invaluable analysis resources and administrative support beginning the first active SoW day. The spaces and resources were ideal for this type of review.

2.4 The Investigative Process

The vast majority of this work took place in SAIT Laboratory. In accordance with our project plan, the investigation began with a short collaborative planning phase. The team met in SAIT Laboratory and spent several days examining code, documentation, and subject reports to understand the problem and to formulate an investigative approach. We then followed the resulting plan that was submitted to the Florida Department of State in accordance with the Statement of Work [1].

As the first objective, the team constructed a flaw list from surveyed literature; the flaw list is given in Appendix A. This list drove the analysis. The team did *not* conduct a comprehensive software review. Rather, we sought to determine if previously documented flaws exist in the target software. It is possible that there are other flaws that were not identified in previous reports; this study would not have identified them because the team was only looking at the question of whether previously identified flaws were fixed. This is different from a comprehensive security review. Despite not actively looking for new flaws, we did identify some and include a discussion of them in the private Appendix C.

The Florida Department of State provided the team fully functional hardware and accessories matching the target software. We utilized these systems to confirm our understanding of the software. We did *not* conduct penetration or red team testing for these systems, as that was outside our charter.

The two target systems are the Diebold Optical Scan and Touch Screen systems that are presently submitted to the Florida Department of State Bureau of Voting Systems Certification. Specifically, we were provided the AccuVote TSX™ version 4.6.5, TSX bootloader version 1.3.6, and AccuVote OS™ version 1.96.8 software. We considered previously-identified flaws in earlier versions of all software for both systems including the AccuBasic interpreters. Additionally, we were provided GEMS 1.18.25 to analyze interaction between this election management system and the two target voting devices.

2.4.1 Limits of this Study

Our analysis examined only those flaws previously reported in the cited literature. We examined the source code and the systems to determine whether or not the reported flaws still exist. Where appropriate, we attempted an attack that would exploit a reported flaw to demonstrate our findings.

Any study involving systems and source code of the complexity of the Diebold Optical Scan and Touch Screen raises questions of completeness: could the investigators have missed problems? We have documented our efforts to allow others to evaluate the thoroughness of our study. When we have found successful attacks, we describe them in sufficient detail in the private appendices to allow others to duplicate them. Where we believe the flaw has been mitigated or fixed, we describe our basis for that assertion. Throughout, we document our assumptions so others may evaluate our results in the context of their environments.

It is important to understand that our conclusions were guided by the source code examined. This means that if the code on a given system does not correspond to the source code we examined, our results may not apply. Further, if the programs that compile, link, load, or install the software, or any libraries or code linked in, disrupt the correspondence between this source and the given system, our results may not apply.

Finally, we do not claim that our results extend beyond the scope of our investigation. We reiterate that the purpose of this report is to evaluate the technical properties of the current systems with respect to our list of previously-identified flaws. The purpose of this report is neither to condemn nor endorse these systems and the findings herein should be considered in the context of the overall election process. We specifically do not contend that these systems are correct or secure beyond the specific results given here. This report is concerned solely with the question posed in the Statement of Work [1] and we do not claim that these results extend to a broader context.

2.4.2 Diebold Known Flaw Details

We analyzed reports discussing previously discovered flaws in the systems [2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12,15]. The resulting flaw list given in Appendix A identifies one hundred and twenty six (126) items and we confined our study to them.

It is also important to note that the items in this list have been rigorously debated by the public and for many of the listed flaws, their proper threat characterization remains unclear and sometimes highly contentious. In our assessment, some of the items are benign. Additionally, several items that made the list do not apply to this study, for example, they apply to a fundamentally different architecture than the software we reviewed. To retain the integrity of the list, we chose to retain them, but do not address them further.

2.5 Diebold Software Architecture

To protect intellectual property, we avoid providing details where these details are not relevant to our findings. We provide the following observations to give context to our findings.

2.5.1 Code Structure

As we expected, the code style and readability varied significantly across the code base. One documented item (#29) identified complicated code as a problem, and there is substantial complicated code throughout the system. Other flaw list items (#29, 30, 32) pointed to poor internal program documentation, such as missing and weak comments. Another item (#31) highlighted design documentation weaknesses. Other than the latter of these, we found the code pretty much as described in the literature. While the documentation we received was much more comprehensive than we expected, detailed design documentation was not available.

2.5.2 Off-The-Shelf Software

As with most modern applications, there are several proprietary components to the reviewed voting systems. While we considered application-operating system interactions and documented flaws relative to bootloaders, we did not independently investigate the proprietary operating system, database system, or driver software. The systems also incorporated non-proprietary components developed openly by the public software development community.

3 Findings

We group our findings according to their related functional areas. We address software components grouped by optical scan firmware, bootloader, interpreter, touch screen firmware, etc. Because many flaws overlap, we cross-reference extensively throughout the document to ensure consistency and to provide context to related flaws.

3.1 Overview

The Team's primary finding is that while we find many improvements in mitigating the vulnerabilities in both the Optical Scan and Touch Screen software that we reviewed, both still retain significant software flaws. In many cases it appears that the vendor attempted to fix these flaws but that the attempted fixes introduced regression faults.

As example of the issues that remain, flaws in the Optical Scan software enable an adversary to introduce an unofficial memory card into an active terminal before the voting (or early voting) period begins. This memory card can be preprogrammed to redistribute votes cast for selected candidates on that terminal including swapping the votes for two candidates. The attack can be carried out with a reasonably low probability of detection assuming that audits with paper ballots are infrequent and that the preprogrammed cards are not detected before use. We implemented this attack in the laboratory and it succeeds despite new protection mechanisms apparently designed to protect against similarly-documented attacks in previous studies.

Many reported flaws were eliminated on the Touch Screen system as well. Nonetheless, we identified and constructed an attack that would allow an adversary to convert official, activated voter cards into smart cards that would enable voters to cast multiple ballots in a ballot-stuffing attack. These cards could be used at any precinct throughout a county. While polling place procedures may mitigate this attack, the attack might evade even rigorous policy enforcement. More damaging is that this attack is not easily correctable. The malicious ballots, whether electronic or paper, would be essentially impossible to identify, so they could not be removed.

That said, we caution the reader that our findings are slanted towards negative results. We focus on flaws that are not completely fixed. Even where we describe flaws that are greatly improved, our focus is on any remaining weakness, as is our charter. Conversely, we do not discuss the many flaws that may have been removed, nor do we describe the structural or general system improvements that we see, though in our conclusion we do identify some cases where flaws appear fixed.

We reiterate that the purpose of this report is to evaluate the *technical properties* of the current systems with respect to previously-identified flaws. Its purpose is neither to condemn nor endorse these systems itself, but rather the findings herein should be carefully considered in the context of the overall election process.

3.2 Prerequisites and Mitigation

As noted above, not all software flaws create vulnerability. In order to expand the context for each flaw, we include a description of circumstances or environments sufficient to exploit the associated potential vulnerability resulting from each remaining flaw.

Additionally, we point out potential procedural or technical processes that could mitigate, reduce, or eliminate the associated vulnerability. Our list of potential mitigation strategies is not meant to be comprehensive, but rather to indicate one direction for addressing the vulnerabilities. As is the nature of computer security, we also recommend that any implementation of a mitigation strategy be subject to a rigorous security evaluation.

Finally, a standard tenant in security is "defense in depth," which suggests that a system should provide adequate security even if individual components fail. We recommend that this principle be considered when evaluating candidate protection mechanisms, whether procedural or technical.

3.3 Prospective Flaws Identified in the Literature

As we noted above, our investigation began with an intense literature review. This Findings Section reports technical information that our analysis revealed about the items documented in the flaw list.

By our analysis, one hundred eight (108) of the one hundred twenty-six (126) flaws that we identified are pertinent to our review. Thirty-seven (37) pertinent flaws were not repaired, thirty-one (31) flaws were improved without complete removal, and forty (40) flaws were removed, corrected, or mitigated.

3.4 Hardware and Physical Security Issues

The flaw list contains fifteen items related to hardware and physical security issues. Our findings suggest that all hardware flaws are either corrected or not applicable to this review, except the one item below.

3.4.1 Diagnostic Mode Not Protected (#55)

An individual with unsupervised terminal access could place an Optical Scan terminal into *diagnostics* mode by simply pressing the two exposed "on" and "off" buttons on the machine and resetting it. From there, she could, for example, connect a device to the serial port and obtain all or most of the data on the memory card as described in Section 3.8.1.1, or reset the machine clock.

Attack Prerequisites: In order to exploit this vulnerability, the attacker must:

1. Have the necessary hardware resources. (Depending on what she wants to do in diagnostics mode, she may need none.)
2. Have a brief period of unsupervised access to an Optical Scan terminal.

Potential Mitigation:

1. Election officials: Rigorously enforce rigid physical access control to optical scan terminals.
2. Vendor: Protect access to the optical scan diagnostics mode or employ sufficient cryptographic protection mechanisms.

3.5 The Signature Flaw

The fulcrum for many of the intended repairs is the vendor's RSA signature. Thus, the weakness we uncovered in this scheme is one the widest ranging issues with the optical scan software and is also important to several remaining issues with the TSX.

The hand-coded RSA signature verification is insecure and signatures generated with the implemented method can be forged. This is true of the signatures on the AccuBasic scripts that are run on the optical scan machines and of the signatures on the operating system images that are to be installed on the touch screen terminal. The ability to run arbitrary AccuBasic scripts on the optical scan machines partially enabled the Hursti attack [5], and the ability to replace unverified boot loaders and operating system images enabled the Princeton attack [4].

The code applies what might initially appear to be a fairly standard RSA signature, using a SHA1 hash, a 2048 bit (or sometimes a 1024 bit) RSA modulus, and a public exponent of three (3). Since the SHA1 digest is 160 bits, the RSA input is *padded with zeros*. To verify the signature, a decode operation computes the modular exponentiation of the signature with the corresponding public key to transform it back into the SHA1 hash of the signed file. However, when the result of the computation is checked against the computed SHA1 hash of the signed file, only the 160 bits that correspond to the SHA1 digest are compared to the hash. The other 1888 bits are not examined. Not examining the other 1888 bits renders this RSA signature variant vulnerable to forgery attacks.

Given the way that signatures are verified, we can forge a signature on any file that has an odd SHA1 hash, i.e. a SHA1 hash with a one in the least significant bit. Since we can almost always add spaces, no-ops, or other data that will not affect the way files are interpreted, we can forge a signature on a message with essentially any semantic meaning we choose. Our code automatically modifies an AccuBasic script to give it an odd SHA1 hash without altering the script's functionality and then forges signature on the script. In its entirety, the code consists of approximately 250 lines of Java, using the standard Java 5 Application Programming Interface (API), and executes in a negligible amount of time. See Inset 1 below for more details of the signature verification process and an overview of the attack.

Attack Prerequisites: This functional flaw is not exploited independently, but is utilized to manifest vulnerability from other flaws. In order to exploit this vulnerability, the attacker must:

1. Discover a corresponding application vulnerability protected by the signature.
2. Gain appropriate access and resources to forge the necessary signature (the public RSA key and a commodity PC).
3. Inject malicious data with the forged signature into the application.

Potential Mitigation:

1. Election officials: Rigorously enforce rigid physical access to media and devices.
2. Vendor: Employ a standardized and widely-accepted mode for public key signature generation and verification. Consider, for example, the PKCS #1 RSA Cryptography Standard.

3.6 Optical Scan Memory Card is Not Integrity Protected (#90)

The data on optical scan memory cards is neither encrypted nor authenticated save the insecure signature on the AccuBasic script, which resides on the card. This vulnerability leads to many potential attacks. In Section 3.8.1.4 below, we describe how we constructed several exploits that manipulate the vote counts on a memory card during the voting day.

With an understanding of the checksums on the memory card data, manipulating any of the unauthenticated data is not difficult. Thus, we further constructed exploits that could be carried out by an adversary with access to a memory card prior to election day and also note other possible threats.

The signature is computed as a plain RSA signature on the SHA1 digest of the message. (We inferred this generation procedure from the signature verification code.) Specifically, let n be the public RSA modulus, d be the private RSA key (exponent), 3 be the RSA public key (exponent), and M be the message to be signed. Furthermore, let $x : a - b$ denote the a through b^{th} bits of x where bit 0 is the least significant bit of x. Then the signature σ is computed as follows:

$$H = \text{SHA1}(M)$$

$$\sigma = H^d \bmod n$$

To verify that σ' is a valid signature on message M', the code computes and checks the following:

$$H' = \text{SHA1}(M')$$

$$H'' = \sigma'^3 \bmod n$$

If the least significant 160 bits of H'' match H', then report VERIFICATION SUCCEEDED. Otherwise, report VERIFICATION FAILED.

If a σ' can be found for many messages M' without knowledge of d such that the verification succeeds, then signatures can be forged. Two steps are sufficient to produce such forgeries: 1) Modify the message M' to an M'' such that SHA1(M'') is odd. 2) Construct σ' such that SHA1(M'') = ($\sigma'^3 \bmod n$) : 0 – 159.

The first step requires simple insertion of characters that will not change the meaning of M' such as trailing spaces, no-ops, and other possible alterations that will not significantly affect the way the message is interpreted.

For the second step, we developed a simple algorithm for computing the forged signature itself, which we have omitted from this report. We confirmed the algorithm's correctness and efficiency by forging signatures on many custom AccuBasic scripts that properly verified and executed on the optical scan machine.

Inset 1. Overview of the RSA Signature Verification Process and Attack.

By manipulating the physical ballot data stored on the memory card, we were able to:

1. Swap two candidate's vote counters.
2. Cause all votes in a race to be tallied for a single candidate of our choice.
3. Cause all votes for one or more lesser-known candidates to tally for a selected candidate.
4. (Many alternative distributions.)

Aside from the exploits we conducted in our lab, the lack of authentication of card data presents several other potential threats. The memory card contains data that is used for a variety of purposes throughout the AV-OS firmware. Because identifying and preventing all possible buffer overflows is an exceptionally difficult task, there is little assurance that a buffer overflow exploit could not be conducted through the manipulation of memory card data. Such an exploit could, for example, allow an attacker to alter optical scan machine memory or enable her to execute code of her choice and essentially gain complete control of the machine.

Finally, the memory card for the optical scan machines contains many memory pointers. The pointers are used to address candidate vote counters, ballot data, scripts, audit logs, and other structures. There is no guarantee that an adversary would not be able to cleverly manipulate these pointers in a way such that she could change vote counts or otherwise maliciously alter the functionality of the machine.

As with other attacks on the optical scan terminal, these attacks might be detected by aggressive auditing or during a recount. Once detected, they could be corrected by properly counting the paper ballots.

Attack Prerequisites: In order to exploit this vulnerability, the attacker must:

1. Have the requisite knowledge and skills.
2. Acquire a current memory card or complete knowledge of memory card structure and the ballot and the equipment necessary to write to the card.
3. Gain sufficient unsupervised access to the terminal to replace the card before, during, or after the election but before the counts are sent to the central server.

Potential Mitigation:

1. Election officials: Rigorously enforce rigid access to media and equipment to limit exploit opportunity.
2. Vendor: Authenticate and verify all data on the AV-OS memory card whenever it is accessed. Since the AV-OS is very limited in computational power, one efficient way of doing this may be the following, although all solutions we are aware of have some security weaknesses if secure hardware is not used: a.) Create a random, symmetric authentication key during some election initialization step, and store the key in inaccessible, persistent storage. b.) After initially verifying a public key signature on the memory card, use the generated key to update a Message Authentication Code (MAC) on the entire contents of the memory card and a counter value. c.) Increment the counter and update the MAC every time the machine writes to the memory card. (The counter should not be stored on the card.) d.) Verify the MAC with the current counter every time the card is accessed. e.) At the end of the election, sign the contents of the memory card with a unique private key. (If secure hardware is not used to store the key, this scheme will have fundamental security limitations, but the vendor can attempt to store it the most inaccessible location possible and encourage election officials to save the key there for the minimal necessary amount of time.)

3.7 AV-TSX Issues

3.7.1 AV-TSX Firmware Faults

3.7.1.1 Cryptographic Key Management (#21)

Two 128-bit AES keys are used for all authentication and encryption.

The "data encryption key" is used for almost all authentication and encryption purposes, including encrypting and generating the message authentication codes for the stored electronic ballots (described in Section 3.7.1.10) and generating the message authentication codes for the election database file (described in Section 3.7.1.8). It is also used to key the hash that stores the supervisor PIN (described in Section 3.7.1.4).

The "system key" is pivotal to key management. It is used to encrypt the file where the data encryption key is stored along with the smart card passwords ("keys" and "magic numbers") and encrypting audit logs. See Section 3.7.1.3 for the specific ways the smart cards are authenticated and see Sections 3.7.1.8 and 3.7.1.10 for the specific usage of and issues with the keys with the election database file and the stored voter ballots respectively.

The system key is generated by computing an MD5 hash of the machine serial number. Its value is never changed after generation. Since the machine serial number is public, the system key is also

essentially public. Anyone who knows this procedure can generate the system key and can access anything it protects, including the data encryption key and anything that it is used to encrypt.

The data encryption key is loaded in one of two ways. First, it can be read from a locally stored file. As noted earlier, the key file is encrypted under the publicly computable system key, so it is an open file to anyone who knows the encryption procedure (which is standard and well known).

Alternatively, the data encryption key may be loaded from a "security key card". The security key cards are insecurely protected as described in Section 3.7.1.3 (the same as all other smart cards), which allows anyone to read all data from them. They also initially use the default password and always use a fixed magic number (see Section 3.7.1.3 for a description of the magic number). Hence, security key cards should also not be used for storing secret information like keys for non-negligible amounts of time.

Above we noted that the same system uses the same cryptographic key for multiple purposes, including both encryption and message integrity protection. As a general principle, such key-reuse is strongly discouraged by the cryptographic community since the reuse of keys can introduce vulnerability.

Attack Prerequisites: This flaw is not independently exploited; rather it is exercised as a component of an attack on another voting application. In order to exploit this vulnerability, the attacker must:

1. Gain access to the machine serial number and compute an MD5 hash.
2. Gain access to a memory card.
3. Or alternatively, gain access to a current security key (smart) card.

Potential Mitigation:

1. Elections officials: Generate the keys on the security key cards immediately before loading the keys onto the machines and then securely delete them as soon as practicable.
2. Vendor: Use accepted key management techniques and public key cryptography.
3. Vendor: Use secure hardware private key storage and operations.

3.7.1.2 Memory Card Update File is Unprotected (#42, 49, 51)

The RABA Study [11] identified attacks that were allowed by modifying unauthenticated and unencrypted files stored on the memory card. Most files on the memory card are encrypted under the present firmware version. However, the file assure.ini remains unencrypted and unauthenticated and is subject to malicious manipulation.

Given the lack of integrity protection on the assure.ini file, an attacker who could remove the card from the TSX terminal could modify the file to set the machine into pre-election mode. While the terminal is in this mode, the attacker could create valid voter cards. If the authentication key necessary to validate voter cards is the same across precincts, as we understand to be common practice in Florida, these cards could also easily be modified to be used at any other precinct within a county. In some cases modifications may not be required. Therefore, the scalability of this exploit could go far beyond one precinct and affect a whole county. We implemented and demonstrated this attack in our lab.

Attack Prerequisites: In order to exploit this vulnerability, the attacker must:

1. Have relevant partial knowledge of the assure.ini file layout.
2. Provide a supply of appropriately formatted smart cards.
3. Gain sufficient access to a touch screen machine and its memory card. The number of voter cards the attacker could produce is dependent on the amount of time she has with the machine. Note that machines can also be taken off their stands and operate for a significant amount of time on battery power.

Potential Mitigation:

1. Election officials: Rigorously enforce rigid physical access procedures for all electronic voting terminals.
2. Vendor: Authenticate the assure.ini file, preferably with a public key signature. Note that encryption on its own may still not provide integrity protection for this file.
3. Precinct logs might, in some cases, detect this attack after the fact. However, even if the ballot stuffing attack is detected, there are no mechanisms built into the software for identifying the malicious ballots and recovering from the attack.

3.7.1.3 Smart Card Authentication Uses Only a Hard Coded Password (#13, 40, 41, 69)

The data and smart card passwords can now be set by election workers. Nevertheless, the implemented authentication protocol is not secure, allowing an attacker to create counterfeit, validating smart cards, including voter cards.

When a smart card is inserted, the terminal sends a "smart card key" to the card. Thus, an attacker can obtain it by inserting a custom smart card into the machine and recording the received key. Once the attacker has the key, she can pass it to her own machine (such as a PDA), which she can then authenticate as a legitimate voting terminal to a good smart card. At this point, she can read the "magic number" (essentially a global, legitimate smart card password) from the smart card and can use it to forge smart cards of her choice. Ultimately, the attacker only needs the magic number and not the key since she can make cards that always respond to key verifications with success messages, regardless of value.

Attack Prerequisites: In order to exploit this vulnerability, the attacker must:

1. Have knowledge and skill to accomplish the attack.
2. Have appropriate equipment.
3. Have voter access to the terminal.

Potential Mitigation:

1. Election officials: Carefully monitor suspicious smart card handling, including times when cards are in pockets.
2. Vendor: Use an established secure smart card authentication protocol that uses a smart card's ability to compute cryptographic operations. (Several such secure protocols have been studied and established.)

3.7.1.4 Supervisor PIN is Not Cryptographically Protected (#14, 69)

The supervisor PIN is now stored on supervisor smart cards as a keyed hash of the actual PIN. Specifically, the PIN is concatenated with part of the "data encryption key" (the first 64 bits), and an MD5 sum is computed over the resulting string. The first 4 bytes of the MD5 are stored on the smart card.

The most significant weakness of this approach again concerns the key management of the "data encryption key" as described in Section 3.7.1.1. The key can be compromised by an adversary with sufficient access to a voting terminal, and an adversary with it can find the PIN using a simple brute force computation.

Again, the act of using the same key for more than one purpose is generally considered poor practice within the cryptographic community. Moreover, the input to the hash function is 64 bits of the 128-bit data encryption key. Using only 64 bits of the 128-bit AES key in this manner may allow an adversary to recover the data encryption key significantly faster than exhaustive search.

Attack Prerequisites: In order to exploit this vulnerability, the attacker must:

1. Gain unsupervised access to a voting machine or security key card.
2. Obtain the "data encryption key". (See Section 3.7.1.1).
3. Gain access to a supervisor card.
4. Read the PIN from the supervisor card. (See Section 3.7.1.3.)

Potential Mitigation:

1. Election officials: Enforce strict access to voting machines, security key cards, and supervisor cards.
2. Vendor: Store the PIN in protected memory on the smart card and use the smart card's ability to compute cryptographic operations to securely verify the PIN.

3.7.1.5 Insecure Storage Mount (#15)

The storage device is mounted by name only, and "\Storage Card" is the directory that Windows CE normally mounts the storage card to. However, this name is also a legitimate standard directory name that may not be associated with the intended purpose. Hence, this method of assessing whether or not the memory card is present is not secure. As one simple example, an attacker could exploit this vulnerability to force the terminal to save votes to an unexpected location, such as one off of the memory card.

Attack Prerequisites: In order to exploit this vulnerability, the attacker must:

1. Gain access to a touch screen voting terminal.
2. Create a "\Storage Card" directory in the Windows CE installation on the terminal.

Potential Mitigation:

1. Election officials and vendor: Ensure that voting terminals are not in debug mode
2. Election officials: Rigorously enforce rigid physical access procedures for all electronic voting terminals.
3. Vendor: Verify that the FILE_ATTRIBUTE_TEMPORARY attribute is set for the directory [8].

3.7.1.6 System Configuration Information is Unprotected (#16)

A large portion of the system configuration is still stored in the system registry and can be altered by directly modifying the registry. The file settings.h defines several large macros for accessing the registry, and appsettings.h defines many specific application settings and functions for accessing those settings using the macros of settings.h. These configurations include informative data such as machine serial number and precinct number and security critical settings such as whether or not the machine should use SSL for network connections.

Attack Prerequisites: In order to exploit this vulnerability, the attacker must:

1. Have access to a touch screen terminal, memory card, and the memory card slot.
2. Have skill and knowledge to load malicious code that would change the system registry or to load a new operating system image with an altered registry, both of which require knowledge of the bootloader process.

Potential Mitigation:

1. Election officials: Rigorously enforce rigid physical access procedures for all electronic voting terminals.
2. Vendor: Authenticate this configuration data.

3.7.1.7 Protective Counter Stored in a Mutable File (#17)

The "protected" counter is read from a mutable file located in the "system directory", which is specified in a registry key. No cryptographic or other checks are made. The counter is written back to the same mutable file. Again, no cryptography or additional precautions are used.

Attack Prerequisites: In order to exploit this vulnerability, the attacker must:

1. Have access to a touch screen terminal.
2. Have skill and knowledge to manipulate or destroy the counter.

Potential Mitigation:

1. Election officials: Rigorously monitor physical access to voting terminals.
2. Vendor: Use secure hardware with a monotonic counter or non-erasable storage to provide a reasonable guarantee that an attacker could not undetectably roll back the counter.
3. Vendor: Use hash chains and other techniques from cryptographically secure audit logs to help prevent an adversary from rolling back the counter prior to the point of compromise.

3.7.1.8 Ballot Definition File is Unprotected (#18)

The ballot definition file is now authenticated with a home-grown Message Authentication Code (MAC). The software uses an encrypted hash value for message integrity protection, which mitigates this vulnerability but does not eliminate the problem. The employed message integrity computation is a non-standard construction that is well-known to be insecure under the standard definition of integrity for message authentication codes; details are given below. The code in question authenticates the file by computing an MD5 hash over the file's contents and then encrypting the hash using AES in ECB mode.

There are two main concerns that result from this authentication method. The most direct concern regards key management. The AES key used is stored in an encrypted file whose encryption key is deterministically constructed from the machine serial number (see Section 3.7.1.1). This gives an attacker the ability to modify the ballot definition file and generate a new, valid authenticator.

The second concern arises from the fact that MD5 is a deprecated hash function and is no longer considered secure [13]. Researchers have developed efficient methods for finding collisions in the function. Thus, even assuming sound key management, there is no guarantee that an attacker could not create two ballot definition files that yield the same hash and hence both validate using the same MAC, possibly leaving the ballot definition file susceptible to attack by someone with control of the ballot definition. Furthermore, the existence of collision-finding attacks against MD5 means that the message authentication scheme used in this software does not meet the standard definition of integrity for such cryptographic objects. CMAC-AES and HMAC-SHA256 are two common message authentication schemes of choice.

Finally, note that since stored votes are only associated with a candidate number and not a name (see Section 3.7.1.12), the ability to create custom ballot definition files allows one to alter or switch candidate names without any record in the vote counts or electronically stored ballots.

Attack Prerequisites: In order to exploit the most direct vulnerability above, the attacker must:

1. Know the machine serial number and the key construction.
2. Understand how the ballot definition file authenticator is computed.
3. Know the memory card file structure.
4. Gain unsupervised access to the voting terminal.

Potential Mitigation:

1. Election officials: Rigorously enforce rigid media protection control procedures.
2. Vendor: Sign and verify the file contents with a secure signature scheme.

3.7.1.9 Impersonate a TSX Terminal to GEMS (#20)

This issue is discussed in detail in GEMS Section 3.10.3 below.

3.7.1.10 No Integrity Protection of Stored Electronic Ballots (#22, 23)

The authenticity protection mechanisms used here are similar to the protections used for the election database file as explained in Section 3.7.1.8, and they share the same issues.

Using the vender's terminology, a "signature" is generated on each ballot/ResultRecord. The code MD5 hashes the ResultRecord and then uses AES in ECB mode to encrypt the hash using the "data encryption key" to produce the authenticator, or Message Authentication Code (MAC).

The most significant problem here, as with the election database file (Section 3.7.1.8), is that there are no clear means of securely managing the key used for the message authentication code ("signature") and encryption. Additionally, given historical attacks against other systems that reused cryptographic keys for multiple purposes, we recommend that the same key not be used for both generating the message authentication code and for encrypting the message.

Furthermore, as also discussed in Section 3.7.1.8, MD5 is deprecated and researchers have developed efficient methods for finding collisions in the function. The existence of collision-finding attacks against MD5 implies that the message authentication scheme used in this software does not meet the standard definition of integrity for such cryptographic objects. CMAC-AES and HMAC-SHA256 are two common message authentication schemes of choice.

Finally, given the use of CBC mode for encryption, the initialization vector used for each encryption should be unique and unpredictable.

Attack Prerequisites: In order to exploit the most direct vulnerability above, the attacker must:

1. Know the machine serial number and the key construction.
2. Understand how the electronic ballot authenticators are computed.
3. Know the memory card file structure.
4. Gain unsupervised access to the voting terminal or memory card before the electronic ballots are tabulated or transferred.

Potential Mitigation:

1. Election officials: Rigorously enforce rigid media protection control procedures.
2. Vendor: Use public key encryption and signatures to encrypt and authenticate each electronic ballot.

3.7.1.11 Ballots are Stored Sequentially (#26, 27)

When a ballot is cast, it is encrypted and appended to the ballot files in the primary and secondary storage directories. Thus, the ballots are stored in the order that they are cast and anyone with the encryption key could correlate votes with voters.

Furthermore, a timestamp, with second granularity, is also stored (encrypted) with each ballot. So, not even subsequently shuffling the order of ballots will prevent someone with the encryption key from linking votes with the voters who cast them.

Attack Prerequisites: In order to exploit this vulnerability, the attacker must:

1. Have unsupervised access to the voting machine
2. Have access to the proper encryption key (see Section 3.7.1.1).

3. Know when voters vote or the order in which they vote.

Potential Mitigation:

1. Election officials: Rigorously enforce rigid key management practices.
2. Vendor: Store the ballots in a non-serial order and without a timestamp.

3.7.1.12 Candidate Information is Not Stored in the Results File (#19)

The cast and electronically stored ballots ("ResultRecords") do not contain any information about candidates. For all non-write-in candidates, each ballot only stores, for example, that candidate number 1 received a vote, candidate number 5 received a vote, etc. Thus, if the names on the screen do not correspond to what is expected, votes will be counted for the wrong candidates. Furthermore, an attack that could alter the presentation of the names without trace, for example by temporarily modifying the ballot definition file, would be particularly effective since there would be no way to detect the attack.

Attack Prerequisites: In order to exploit this vulnerability, the attacker could:

1. Design and authenticate a custom ballot definition file with altered candidate names (see Section 3.7.1.8) or find another exploit.
2. Gain access to a terminal memory card or an election office where ballot definition files are stored.
3. Overwrite the legitimate ballot definition file with the custom one.

Potential Mitigation:

1. Election officials: Rigorously enforce access control to voting machines and ballot definition files.
2. Vendor: Store the displayed names or other explicit information corresponding to every vote with each electronically cast ballot.

3.7.1.13 Audit Logs are Not Cryptographically Protected (#28, 49)

The logs are encrypted and authenticated the same way as the electronic ballots, using the "system key", which is insecure. (See Section.3.7.1.10) Hence, they are also susceptible to viewing and modification by an adversary. Among other things, such an ability may allow an adversary to erase record of an attack without detection.

3.7.1.14 Data is Neither Authenticated Nor Encrypted Over the Communication Link (#25, 35, 45, 46)

The TSX firmware now supports SSL protection of its GEMS connections, though its use is optional. This issue is discussed in detail in GEMS Section 3.10.3 below.

Additionally, the SSL pseudorandom number generator is seeded with poor entropy, based on tightly bounded clock measurements. Cryptographic protocols often provide degraded security when the random numbers are not securely generated. To mitigate potential concerns, the vendor could seed the cryptographically secure pseudorandom number generator with sufficient entropy.

3.7.2 AV-TSX Bootloader Faults

3.7.2.1 Bootloader Automatically Replaces Itself (#3, 4, 6, 11)

There are now two processes to update software in the TSX. One of these automatically updates the unsigned bootloader with a file named "eboot.nb0" if it is found on the memory card. This method is conditionally compiled. However, in the software version that we are analyzing, the code is included.

In this approach to updating, the bootloader does not perform any kind of cryptographic authentication of the replacement bootloader found on the memory card. This vulnerability is mitigated by the requirement that the machine be in "debug mode" for the update to occur. An attack loading a custom, malicious bootloader is immediately possible if the terminal is delivered to the polling place already in

debug mode. Otherwise, an attacker must open the case and move a hardware switch to enable this attack.

While enabling debug mode can only be done with physical access to the hardware, this protection is not as strong as would be provided by requiring a signed bootloader or another strong access control method.

The other software update method that appears to be implemented more recently conducts software updates through the supervisor account when the normal voting machine software is running. Nevertheless, we detected a shortcoming in the supervisor PIN verification process (see Section 3.7.1.4 above).

Attack Prerequisites: In order to exploit this vulnerability, the attacker must:

1. Have a modified memory card with a tested bootloader and
    a. Gain access to a terminal in debug mode or
    b. Obtain a compromised supervisor PIN.

Potential Mitigation:

1. Elections officials must rigorously ensure that TSX terminals are not routinely in debug mode.
2. Vendor: Remove the automatic filename update code from the source.
3. Vendor: Fix the supervisor PIN problem (see Section 3.7.1.4 for the PIN problem description).

3.7.2.2 Bootloader Automatically Replaces Local Operating System (#7)

The flaw has been changed, but still exists in a different form. If the bootloader finds a file named "nk.bin" (or some other nk.* files) on the terminal's memory card, rather than installing it as the new permanent operating system, it boots it as the current one. Once again, the file is not authenticated. This only occurs when the machine is in debug mode.

Attack Prerequisites: In order to exploit this vulnerability, the attacker must:

1. Generate a Windows CE operating system image (Microsoft provides tools to do this) with malicious voting software.
2. Gain access to a terminal in debug mode.

Potential Mitigation:

1. Election officials: Rigorously enforce rigid media protection control procedures.
2. Vendor: Sign and verify the file contents with a secure signature scheme.

3.7.2.3 Bootloader Automatically Runs .ins Files on the Memory Card (#8)

The bootloader also uses .ins files to install new operating system images. Specifically, if the machine is in debug mode and the bootloader finds the file "avtsx.ins" on the memory card, it will attempt to install a new operating system image contained within the file. The .ins files are now signed with an RSA signature variant, but the signature is not implemented properly and can be forged. The details of the signature and its weakness are described in Section 3.5. An attacker could therefore create a Windows CE image of her choice that properly authenticates. Our code for forging signatures on .abo scripts can be ported to do the same for .ins files because they are authenticated by the same faulty verification code.

Attack Prerequisites: In order to exploit this vulnerability, the attacker must:

1. Generate a custom Windows CE operating system image (Microsoft provides tools to do this) with malicious voting software and insert it into an "avtsx.ins" file.
2. Gain access to a voting terminal in debug mode or gain several minutes of access to any terminal.
3. Insert a new memory card with the custom "avtsx.ins" file.

Potential Mitigation:

1. Election officials must ensure that debug mode is disabled on all voting terminals.
2. Election officials must strictly monitor access to voting terminals.
3. Vendor: Properly cryptographically sign and verify all .ins file before running them.

3.8 AV-OS Software Issues

3.8.1 AV-OS Firmware Faults

3.8.1.1 Leaks Memory Card Contents (#56)

In this vulnerability, an attacker uses a designed machine function in precisely the way that it was intended. That is, the attacker can copy the memory card contents to her own laptop by connecting the laptop to the optical scanner serial port or phone line, turning the machine on, entering diagnostic mode by simultaneously pressing the "yes" and "no" buttons, and selecting a menu option to dump the memory card's contents. The attacker's only technical contribution is establishing a routine terminal emulation to synchronize the laptop with the Optical Scanner, which is a standard function. Alternatively, the modem can also be used to dump the data.

We connected the optical scanner to a Windows XP computer and we were able to dump the contents of the card onto the computer using HyperTerminal. Though our experiments only dumped the first 4kB of the 128kB memory card, the data included all of the ballot definition and scripts.

Attack Prerequisites: In order to exploit this vulnerability, the attacker must:

1. Have the knowledge of the exploit.
2. Have unsupervised access to a terminal.

Potential Mitigation:

1. Election officials: Rigorously monitor access to the optical scan machine.
2. Vendor: Allow memory card dumps only from supervisor mode.

3.8.1.2 Supervisor PIN Not Cryptographically Protected (#57, 92)

The supervisor PIN was removed from the source code and an encoded version is now placed on the memory card. When a user wants to enter supervisor mode, the following authorization protocol takes place:

- The shuffled PIN and a decryption key are read by the terminal from the election header fields on the memory card.
- The user enters her PIN on the machine.
- The shuffled PIN is decoded using the decryption key and compared against the PIN entered by the user.

The remaining weakness is that in this protocol, the key that reveals the PIN is also on the memory card, so anyone with access to a card and reader and knowledge of the shuffling algorithm can extract the shuffled PIN and the key and thus can decipher the supervisor PIN.

Attack Prerequisites: In order to exploit this vulnerability, the attacker must:

1. Have unsupervised access to a machine to dump a current memory card or access to a memory card itself.
2. Have the necessary equipment to read from the memory card.
3. Have knowledge of the key shuffling algorithm.

Potential Mitigation:

1. Election officials: Exercise rigorous control procedures over voting terminals and removable media.

2. Vendor: Securely encrypt and authenticate the PIN and deliver the key to the machine through a medium that is as inaccessible as possible to most people. Ideally, the decryption key should be stored in secure hardware, and unless secure hardware is used, there is a bootstrapping problem that creates a fundamental limitation in the privacy of the PIN.

3.8.1.3 No Authentication Between GEMS and the Terminal (#58)

The primary concern with this vulnerability is if the devices were to connect across the Internet. The vendor strongly states that no such connection occurs. If the connectivity is through a serial connection with trustworthy technicians utilizing the connection, the potential impact is largely negated.

However, if the connection is via a modem, a man in the middle attack may be possible. The vendor wrote code to mutually authenticate the GEMS server and optical scan terminal, employing a home-grown encryption algorithm. Much like the protection of the Supervisor PIN, described in Section 3.8.1.2 above, critical components are provided in encrypted form within messages that contain the decryption key. This practice does not provide the intended security properties. More specifically, the protocol relies on the exchange of a password that is encrypted using weak encryption functions and the keys are exchanged in the clear at the beginning of each handshake, leaving the protocol vulnerable to an eavesdropping attack.

Attack Prerequisites: In order to exploit this vulnerability, the attacker must:

1. Have access to the memory card or a mechanism to eavesdrop on the communications.
2. Have knowledge of the communication protocol.

Potential Mitigation:

1. Elections officials: Never connect to the AV-OS terminals to the Internet or other untrusted networks.
2. Vendor: Remove code supporting networking connections or require use of a secure, mutually authenticated protocol, such as SSL.

3.8.1.4 Attacker Can Hide Preloaded Votes (#59, 85, 89, 90, 94, 95, 96)

The original attack has been largely mitigated. However, it is still possible to load selected vote counts on a memory card that the terminal will recognize if inserted and it is still possible to forge AccuBasic Interpreter scripts. We constructed four exploits of this vulnerability in the lab.

1. We prepared a memory card with preloaded votes and inserted it into the terminal after the zero report printed. The machine accurately displays the number of ballots that have been cast (or loaded), which could be detected by a conscientious precinct clerk or poll worker. The attack requires less than a minute of unfettered access to the terminal at the beginning of the voting day.
2. In our second exercise, we extracted a memory card from the terminal after several votes had been registered, modified the counters to redistribute votes, but also to maintain the same total vote count, and reinserted the memory card in the terminal. This process took several minutes of access to the terminal, and if it were accomplished in a real election, it would be detectable by a recount of the paper ballots.
3. In the third exploit, we prepared a memory card with a predetermined vote count and waited until the number of votes on the card were normally registered on the terminal. We then replaced the official memory card with our forged card. The machine display reflected the correct, expected number of votes cast and the precinct count printout showed the specific candidate votes that we injected. This attack took only a few moments access to the terminal, though it demands that the memory card be inserted when the appropriate number of ballots have been cast. If this attack were conducted by a poll worker who could switch cards at the precise point when the machine count

matched the malicious card's vote count, it is unlikely that the attack would be detected unless a recount occurred. In a recount, the count would be corrected based on the paper records.

4. We constructed an interpreter script that disregards the counters on the memory card and prints the counts that we preload. This attack would normally be detected and corrected when the electronic vote count is transferred to the SoE. If the discrepancy between the two electronic counts is noted, the correct count could then be confirmed by a recount of the paper ballots.

There are two software issues related to this vulnerability. The first is the RSA signature described in Section 3.5 above. As we demonstrated in the lab, without knowing any key, we can forge a signature on essentially any functional interpreter script that we desire. This issue applies to bullet 4 above.

The second issue concerns the fact that vote counts on the memory card are not properly protected. This allowed us to edit fields to change votes with our only challenge being to appropriately set error correcting checksum values. This issue applies to bullets 1, 2, 3, and 4 above.

Attack Prerequisites: In order to exploit this vulnerability, the attacker must:

1. Have unsupervised access to the optical scanner.
2. Have access to uncommon memory cards and equipment to read and write them.
3. Have significant computer skill and access to the proper ballot definition.

Potential Mitigation:

1. Election officials: Restrict access to removable media, particularly memory cards.
2. Vendor: Authenticate the contents of the memory cards including vote counts. (More details on this are suggested in Section 3.6.)
3. Vendor: Employ secure audit logging techniques as developed within the security and applied cryptography literature.

3.8.1.5 Vote Counters Are Not Directly Checked for Overflow (#85, 89, 95, 96)

Candidate ballot counters are protected from overflow by a total ballot counter. This is an imprecise, and apparently an overly strict approach that detects when a machine has reached the limit for overall votes. Since individual candidate vote counters can hold as many votes as the total ballot counter, this approach appears to prevent overrunning all vote counters.

This check depends on an invariance between the protected and protecting variables. It assumes that votes are always recorded through the voter interface and processed by the firmware that ensures the necessary consistency. This works if the counts start at zero and votes are always recorded through the intended paper ballot interface. However, this may not be the case. Specifically, Hursti programmed a memory card to effectively store a negative vote count by storing a very large vote count, which then overflowed [5], though this did not cause the total ballot count to overflow.

Other consistency checks, such as whether the total number of votes in a race equals the sum of the votes for each candidate in the race are also now used, and may prevent a possible exploit of this flaw.

While we do not see any immediate attacks caused by this issue, it invites consistency attacks, for example, that may attempt to subtract votes from a candidate with zero votes by manipulating the memory card. There are mechanisms that appear to prevent attacks that we examined in the code. However, defensive programming practice suggests implementing overflow protection at the point where the variable is computed.

Attack Prerequisites: In order to exploit this vulnerability, the attacker must:

1. Find a malicious means of overflowing a vote counter.
2. Acquire a memory card and knowledge of its structure.
3. Create a memory card such that the appropriate vote counter would overflow.

4. Insert the memory card into an optical scan machine.

Potential Mitigation:

1. Election officials: Strictly monitor access to optical scan machines and memory cards.
2. Vendor: Directly verify that vote counters do not overflow every time they are incremented.

3.9 AccuBasic Interpreter Faults

3.9.1 Error Checking is Inadequate (#67, 68, 81, 82)

The interpreter does not provide descriptive prompts as to errors that were encountered. Most of the prompts are along the lines of "worked/didn't work". Once the main interpretation has started, there seems to be no printing of error messages to indicate problems.

While we did not develop an example attack, this design weakness might make an attack indistinguishable from a random error if such an attack were employed.

3.9.2 Error Codes Returned by the AV-OS System are Ignored (#83)

A specific function provides the entry point to the AccuBasic interpreter and returns a status code indicating the success or failure of script interpretation. In all instances within the AV-OS code where this function is called the return value is ignored.

As above, we did not identify any direct exploits resulting from this flaw, but the lack of error checking may make it easier to execute other attacks without detection..

3.9.3 AccuBasic Scripts Can Be Misused (#62, 86, 87, 97)

The interpreter allows AccuBasic code to perform conditional operations based on comparisons of data including vote counts, time, date, candidate names and any other data to which it has access. This may enable an attacker to hide exploits by presenting the user with conditional information.

Additionally, AccuBasic scripts can interact with the user through the terminal's LCD screen. This may allow a malicious script to use social engineering techniques to enable attacks. For example, an attacker may preload a script that prints a message such as "Bad memory card. Please insert another" in order to accomplish the memory card swapping attack described in Section 3.8.1.4 above.

To illustrate this flaw at a basic level, we wrote AccuBasic scripts that print text of our choice and that also passed signature verification; for additional details see Sections 3.5 and 3.8.1.4.

3.9.4 Public Key Hard-Coded into the Source (#91)

Embedding the public key in the source code does not pose a direct security problem. However, having the public key hard-coded into the source code prevents the vendor from routinely changing the public/private key pair. On the other hand, hard-coding the public key also prevents an adversary from easily changing it. The particular trade-offs between these two approaches should be evaluated in the broader context of election systems.

3.9.5 Unchecked String Operation: Allows Overwrite of Stack Memory (#113, 114)

The vulnerable string operations for these two flaws still exist in the code as described in the VSTAAB report [2]. However, the repairs made to correct other flaws mitigate the vulnerability introduced by these flaws. So the flaws are still there, but the corresponding vulnerability (based on another flaw) seems to be fixed, as we report in Table 1 below. Due to the level of detail involved, we defer further discussion to the private Appendix B.

3.10 GEMS Server Faults

3.10.1 AccuBasic Scripts are Not Authenticated on the GEMS Server (#118)

GEMS itself does no checks to the AccuBasic byte code (.abo) because they rely on the (problematic) RSA signatures (see Section 3.5) on the OS machine to verify the AccuBasic code. GEMS is apparently backwards-compatible with versions of the AV-OS/AccuBasic that did not contain signatures. Any .abo code could be placed on the GEMS server and selected to send to the AV-OS if there are no procedural/physical safeguards. This would be a vector for loading a malicious .abo script, although it is more complicated than simply writing one to the memory card (due to formatting differences in the .abo files), and requires access to GEMS rather than the card itself.

Attack Prerequisites:

1. Have access to the GEMS folder that contains the .abo files and to the option within GEMS itself to select which .abo file will be sent.
2. Create a malicious AccuBasic script, which would require knowledge of the RSA signature flaw. (See Section 3.5.)

Potential Mitigation:

1. Election Officials: Safeguard access to the GEMS server.
2. Vendor: Use a secure signature scheme to sign and verify the AccuBasic scripts.

3.10.2 Password Does not Protect Access to GEMS or Audit Logs (#43, 123)

The GEMS password authentication process can be defeated with a simple attack, given a few moments access to the computer. This attack is publicly known and revolves around the way the password is stored in a Microsoft Access database. We constructed a proof of concept exploit for this vulnerability to confirm its applicability.

The same vulnerability occurs with the audit logs of the TSX.

Attack Prerequisites: In order to exploit this vulnerability, the attacker must:

1. Have knowledge of the exploit. (This attack is described in detail on a public Internet web site.)
2. Have remote or local access to a GEMS server

Potential Mitigation:

1. Election officials: Rigorously protect physical access to GEMS servers.
2. Election officials: Never connect GEMS servers to the Internet or an untrusted network.

3.10.3 Incomplete Implementation of the SSL Protocol (#124)

The use of SSL with the GEMS server is optional. The user is presented a dialog box that allows her to choose both whether or not she wants to use SSL and whether or not she wants to require authentication of the client. Both of these should be made mandatory. The corresponding setting on the TSX client is stored in the machine registry.

When SSL is used, if the option to require authentication is set, GEMS does require authentication from the client. Additionally, the software includes the option to connect using an early version of SSL (V2) that is subject to downgrade attacks.

We also note that the GEMS server does not use SSL when communicating with the AV-OS. Rather, they employ a vendor generated communication protocol for the AV-OS that does not include any SSL calls. (See Section 3.8.1.3 above.)

The TSX will verify that it is establishing an SSL connection with a GEMS server, but not which GEMS server. Similarly, the GEMS server will verify that it is establishing a connection with a GEMS client, but not which GEMS client. While this is an improvement over the SSL implementation evaluated by the RABA Study [11], this implementation is still vulnerable to man-in-the-middle attacks assuming that multiple states or election districts use the same public keys for the Diebold certificate authority (which we assume to be the case since they are hard-coded into the software). For example, an insider from District A with sufficient infrastructure access could use a copy of her GEMS private key and a TSX client's private key to mount a man-in-the-middle attack against the TSX machines and GEMS server in District B.

Attack Prerequisites: In order to exploit this vulnerability, the attacker must:

1. Have knowledge of the weakness and exploit techniques.
2. Have sufficient knowledge of the operating environment.
3. Possess the proper equipment and infrastructure access.

Potential Mitigation:

1. Election officials: Never attach GEMS, AV-OS, or TSX terminals to the Internet or an untrusted network.
2. Vendor: Require client authentication whenever connecting to a remote host.
3. Vender: Verify that the client and server are communicating with the intended parties, not simply a certain class of devices.

4 Non-Pertinent Faults

The team ran automated analysis tools over the code base. The non-pertinent flaws that we identified are contained in the private Appendix C.

Additionally, flaw #44 noted that the code contains third party components. We also found such references in the source that we reviewed. Since we did not find discussions of specific flaws related to third party software in the previous reports that we reviewed, we do not include a discussion of these third-party products in the public portion of this report. We list the third party products that we identified, along with their versions, in private Appendix C.

5 Conclusions

Electronic voting systems offer tremendous opportunity to expand accessibility and reduce costs even in the face of increasing safety and accuracy demands. Conversely, they also offer virtually unbounded opportunity to manipulate elections if they are not properly secured. Code review is one step in this process.

This report presents the background, organization, process, findings, and opinions of our software code review. We conclude with the following summarizing issues.

5.1 Flaw Retention Overview

The team was tasked to determine if flaws identified in the literature remain in the software version submitted for certification. Table 1 reflects our precise answer to this question under the definitions and terminology outlined in Section 2.2.

Applicable Component	No Change	Improved	Fixed
Physical Security	11, 55		48, 50, 51
OS Firmware	56, 62, 64, 92, 93	57, 58, 59	84, 85, 88, 89
OS Interpreter	81, 82, 83, 8, 86, 87, 91, 97, 98	90, 94, 96, 105, 113, 114	80, 95, 99, 100, 101, 102, 103, 104, 105, 106, 107, 108, 109, 110, 111, 112
Bootloader	3, 6	4, 5, 7, 8	1
TSX Firmware	15, 16, 17, 19, 22, 24, 26, 28, 30, 32, 40, 41, 42, 43, 44, 51	13, 14, 18, 20, 21, 23, 25, 27, 29, 31, 34, 35, 45, 46, 49, 69	24, 33, 36, 38
TSX Interpreter	67, 68		65, 66, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79
GEMS	123	118, 124	

Table 1. Flaw Categorization by Flaw Number (See Appendix A)

5.2 The Optical Scan Software

The vendor implemented integrity verification and encryption to address some of the reported flaws. In many cases, unproven designs were applied that left vulnerability where stronger tools could provide measurable protection. We demonstrated several attacks in our laboratory. For one attack, we exploited a flaw in the AV-OS RSA implementation to create a custom AccuBasic script with a forged RSA signature. Our custom AccuBasic script will cause the AV-OS to output incorrect election totals. Many of the remaining vulnerabilities are detectable and correctible through normal elections procedures and would certainly be detected during audits and recounts.

5.3 The Touch Screen Software

The vendor has made improvements to address many of the specific flaws expressed to date. Despite these improvements, many flaws remain in the TSX system. In our opinion, the vendor could make significant additional improvements by employing strong signature protocols and improved key management techniques.

5.4 Removable Media

All removable media associated with electronic voting systems is highly sensitive, and must be protected year-round with the same status and priority as voted ballots. Unsupervised access to an item as simple as a TSX supervisor card and its accompanying PIN allows an individual to create valid voter cards. In the wrong hands, these cards can cause significant damage to a county's vote count validity.

5.5 "Security Through Obscurity" is Not Sufficient to Protect Voting Systems

Two important transitions heightened this realization: (1) the rapid expansion of computer use in

elections; and (2) the explosive growth of computing in general accelerated by the explosion of the Internet. The processes and methods behind these systems are often publicly exposed, as was witnessed when the Diebold Touch Screen source code surfaced on the Internet in 2003 [14]. The exploit described in Section 3.5 clearly illustrates the convergence of theory and practice in electronic voting security. The signature scheme applies well-known cryptographic primitives and is employed in an efficient and intuitively appealing, though home-grown, approach. However, security theory and mathematical analysis revealed a simple, efficient algorithm that allows an attacker to forge essentially any arbitrary message, thus easily subverting this approach.

5.6 Engineering In Security

As with any system, it is most effective and least costly to incorporate necessary components up front rather than adding them later. Just as a house that is designed and constructed with a garage may cost more up front, its overall cost is generally much less than the cost of the house without the garage, and the additional cost of adding the garage later. Worse yet, if the original design did not include provisions for a prospective garage, adding one may not be possible at all.

The issue is similar with voting systems security. Systems designed with security in mind have a much better chance of addressing evolving threats than after-the-fact security response, particularly when prospective security features were not considered or provided for in the original design.

This report amplifies this challenge. A primary issue in our analysis is the impact of regression faults, or faults that occur as a result of modifying a system. In this case regression faults leaked into the system as the result of modifications to correct other faults. Implementing security is much easier if done from the ground up.

5.7 Persistence of Vulnerability

The flaws examined in this study have been published in public reports, several of them over three years ago. Many of these flaws appear in multiple studies, reiterating that they existed. While the vendor has fixed many of these flaws, many important vulnerabilities remain unaddressed, or the attempted fixes leave vulnerabilities in place. Solutions to most of the issues in these systems have been studied in depth and added to the public knowledge base. Although such information is available in a variety of resources, as are knowledgeable professionals, studies have repeatedly shown that home-grown security mechanisms rarely provide the intended security properties. Home grown mechanisms cannot replace systematically developed and time tested security solutions.

5.8 Key Management is a Difficult [Voting System] Problem

It is a well-known that key management is one of the most challenging issues in using cryptography to protect information systems. Whether a system embeds public keys in the source code or attempts to bootstrap a private key separate from application initialization, the challenges of managing keys that enable even mutual authentication are great.

5.9 Issues to Improve Voting System Security Evaluation

With the growth of electronic voting systems, software review is becoming more and more commonplace. We offer the following observations that may help future analysis efforts.

5.9.1 Require Vendors to Deliver a Complete Development Environment with Software

Software review and proof of concept testing are resource intensive processes that demand domain appropriate skills. Conversely, software development environments can vary greatly. In order to reduce both spin up and proof of concept construction time in the review process, states should require vendors to file complete development environments with their systems. The environment delivered

should enable the state, and any review teams, to rebuild the certified binaries. During a review, these should be delivered with the source code to the review team.

5.9.2 Require Vendors to File Design Documentation with the Department of State

Similarly to having development environments, software review can be simplified if accurate, detailed design documentation is available. States should require such documentation and make it available to any software analysis effort.

5.9.3 Independent Security Evaluation

We recommend that voting systems be subjected to ongoing reviews so that flaws can be discovered before an election and before a system is adopted, and proposed fixes can be evaluated for completeness, rather than after an election outcome comes into doubt.

As another efficiency consideration, it is unlikely that all states are fiscally capable of conducting rigorous voting system analysis. Thus, we emphasize that a uniform testing process, such as the one presently under construction by the United States Elections Assistance Commission (EAC) Technical Guidelines Development Committee (TDGC), be adopted and that standard practices for sharing review results among states be implemented.

6 Acknowledgments

As part of our work we used automated source code analysis tools Fortify Source Code Analysis (SCA), made by Fortify Software and Coverity Prevent by Coverity in order to assist with the code review process. Fortify Software donated the tool to us free of charge for use on this project and we thank them for their contribution. We note that one member of the team (Bishop) is on Fortify Software's Technical Advisory Board and Coverity is a SAIT Laboratory partner.

We also acknowledge the major contributions of the ACCURATE Center in this review. We note that the final report's first author is an ACCURATE member and we received significant advice and resources from several ACCURATE members throughout this process, for which we are grateful.

We thank Doug Jones, Avi Rubin, and David Jefferson, who provided several documents that we used in research for this work.

A special thanks to Avi Rubin and Breno de Medeiros for their comments on a draft of this report.

7 References

1 "Software Review and Security Analysis for Diebold Voting Machine Software.", Joint Florida Department of State and Florida State University Statement of Work, May 14, 2007.

2 David Wagner, David Jefferson, and Matt Bishop, "Security analysis of the Diebold AccuBasic Interpreter", Voting Systems Technology Assessment Advisory Board (VSTAAB), University of California, Berkeley, February 14, 2006.

3 Diebold Election Systems, Inc., "Source Code Review and Functional Testing," CIBER, Inc., 7501 South Memorial Pkwy, Suite 107, Huntsville, AL 35802, February 23, 2006.

4 Ariel J. Feldman, Alex Halderman, and Edward W. Felten, "Security Analysis of the Diebold AccuVote-TS Voting Machine", Center for Information Technology Policy and Dept. of Computer Science, Princeton University, September 13, 2006.

5 The Black Box Report, SECURITY ALERT: Critical Security Issues with Diebold Optical Scan Design, July 4, 2005.

6 Harri Hursti, "Diebold TSX evaluation: Critical security issues with Diebold TSX.," Available at http: www.bbvdocs.org/reports/BBVreportIIunredacted.pdf, May 2006.

NF

7 A. Kiayisas, L. Michel, A. Russell, A. A. Shvartsman, "Security Assessment of the Diebold Optical Scan Voting Terminal", UConn VoTeR Center and Department of Computer Science and Engineering, University of Connecticut, October 30, 2006.

8 Tadayoshi Kohno, Adam Stubblefield, Aviel D. Rubin, and Dan S. Wallach, "Analysis of an Electronic Voting System", IEEE Symposium on Security and Privacy, May 9-12, 2004, pp. 27-40.

9 Maryland State Board of Elections, "Response to: Department of legislative services trusted agent report on Diebold AccuVote-TS voting system", Available at http://mlis.state.md.us/Other/voting system/sbe response.pdf, January 2004.

10 Ohio Secretary of State, "Direct Recording Electronic (DRE), Technical Security Assessment Report", Compuware Corporation, 1103 Schrock Road, Suite 205, Columbus, Ohio 43229, November 21, 2003.

11 RABA Technologies. Trusted agent report: Diebold AccuVote-TS voting system. Available at http: www.raba.com/press/TA Report AccuVote.pdf, January 2004.

12 Science Applications International Corporation, "State of Maryland Risk Assessment Report, Diebold AccuVote-TS Voting System and, Processes, SAIC-6099-2003-261, September 2, 2003.

13 Xiaoyun Wang and Hongbo Yu, "How to Break MD5 and Other Hash Functions", EUROCRYPT, May, 2005, pp. 19-35.

14 Bev Harris, "Black Box Voting: Vote Tampering in the 21st Century". Elon House/Plan Nine. July, 2003.

15 Collaborative Audit Committee, "Collaborative Public Audit of the November 2006 General Election". Available at http://urban.csuohio.edu/cei/public_monitor/cuyahoga_2006_audit_rpt.pdf, April 2007.

Appendix A Flaw List

ID	Description	Source	Page #
1	Machine boots into Windows explorer rather than BallotStation if explorer.glb is found on memory card	[4]	5
2	Machine door lock is consistently pickable in <10 seconds	[4]	5
3	fboot.nb0 can be used to load malicious software	[4]	5
4	fboot.nb0 can be overwritten without authentication or integrity checks	[4]	16
5	Denial of Service with PowerOffSystem() API	[4]	16
6	Bootloader replaces itself with eboot.nb0 or fboot.nb0 if found on memory card on boot, no authentication	[6]	5
7	Bootloader replaces OS with nk.bin (or some other nk.xxx's) if found on memory card, no authentication	[6]	7
8	Bootloader "runs" any .ins files in memory card on boot after prompting user, no authentication	[6]	8
9	Back of machine casing comes off and leaves seals in tact	[6]	9
10	Hidden SD memory card slot inside the machine case	[6]	10
11	Jumpers on machine motherboard enable debug features	[6]	10
12	Hidden, voter accessible button on back of case	[6]	11
13	Lack of smartcard authentication (use a hardcoded password)	[8]	9
14	PIN sent from smartcard to terminal in cleartext	[8]	11
15	Insecure file mount, does not ensure writing to removable media	[8]	12
16	Unprotected system configuration file	[8]	12
17	Protective counter stored in mutable file	[8]	12
18	Ballot definition unprotected and unauthenticated	[8]	13
19	Candidate information is not stored in the results file (only vote counts)	[8]	14
20	Impersonate a legitmate voting terminal, no authentication and data can be gathered from ballot definition files	[8]	14
21	Cryptographic key management beyond hard-coded DESKEY	[8]	14
22	DES CBC encryption uses constant 0 for IV	[8]	15
23	No integrity protection of stored vote counts and data	[8]	15
24	No sequence numbers stored with votes (to help detect record deletion, may be problematic with respect to privacy)	[8]	15

25	No integrity checks, authentication, or encryption on votes transferred to the backend server	[8]	16
26	Votes are written serially	[8]	16
27	Bad RNG for randomizing vote order	[8]	16
28	Audit log problems: no consistency for what is logged, does not verify that printer is attached before writing to it	[8]	17
29	Complicated, undocumented code segment	[8]	18
30	Poor change-control process info in code comments	[8]	19
31	Missing design documents	[8]	19
32	Code comments suggesting fixes, not documented further	[8]	20
33	1.06 Parameters and return values of functions are generally not commented	[10]	32
34	1.26 DES encryption key is hard-coded	[10]	36
35	1.27 Data is not encrypted when transmitted over data link	[10]	36
36	1.31 The supervisor's password is hard-coded	[10]	36
37	2.06 System can be locked up by pressing F4 and choosing to open BallotStation.exe	[10]	38
38	2.09 PIN for all smart cards is the same and the factory default (1111)	[10]	39
39	2.14 Several TCP/UDP ports are open	[10]	40
40	Can make counterfeit voter and supervisor cards	[10]	52
41	Can vote multiple times with a card writer	[10]	52
42	PCMCIA cards are not encrypted	[10]	52
43	Microsoft Access database has no password protection (used for ballot definition, audit logs, and tally results)	[10]	52
44	1.16 Software contains third party components	[10]	56
45	Digital certificates only exist at the servers and these are neither signed nor authenticated by the AccuVote terminals	[11]	9
46	No GEMs authentication by the AccuVote-TS terminals	[11]	9
47	AccuVote-TS terminals have two locking bays - all terminal locks are identical	[11]	18
48	With a keyboard, attacker can access options (unnecessary test code) "Save As", "Finish Recording", and "Open" to overwrite results and audit file	[11]	18
49	Only files with cryptographic protection on PCMCIA card are the results file and the audit file (extent of protection, even on these, may not be great)	[11]	18
50	Can attach a keyboard to the terminal and access functionality in the software that allows the attacker to view the entire	[11]	18

	directory tree on the machine's internal memory and on the PCMCIA card		
51	Attacker can load a PCMCIA card with an update file	[11]	18
52	Training does not include an information security component	[12]	5
53	No documentation that identifies the process for maintaining access controls	[12]	7
54	Executes code (Accubasic object bytecode) off the memory card	[5]	1
55	Machine enters diagnostic mode if 2 buttons are depressed when it is powed on, no password needed (allows reinitializing of the machine and its clock, for example)	[7]	6
56	Leaks memory card contents	[7]	6
57	Supervisor PIN not cryptographically protected	[7]	7
58	No authentication between GEMS and the terminal	[7]	9
59	Executes code (Accubasic object bytecode) off the memory card	[7]	4
60	Machine allows multiple feeding of ballots due to feed sensor position	[7]	12
61	AccuBasic interpreter firmware chip is designed to be replaceable	[7]	13
62	Memory card access works as an extension of main memory rather than as a file system	[2]	9
63	Machines do not allow for counting of ballots but only ballot pages (report after 2006 election)	[15]	36
64	Machines do not report data at a machine level of precision, only a precinct level (report after 2006 election)	[15]	36
65	Three types of tokens used to potentially read and modify data in global memory (no specifics given)	[3]	7
66	Bounds on the heap and stack segments do not appear to be checked	[3]	9
67	Error checking is inadequate for identifying and recovering from failures in a damaged or disfunctional environment	[3]	11
68	Error handling makes it difficult to differentiate between malicious activity and failure	[3]	11
69	The contents of the smartcards are neither encrypted nor digitally signed	[11]	17
70	W1 Array bounds violation: Overwrite any memory address with a 4-byte value that the adversary has partial control over. Allows attacker to inject malicious code and take complete control of the machine	[2]	15
71	W3 Input validation error: Choose any memory location and begin executing it as .abo code; could be used to conceal malicious .abo code in unexpected locations, or to crash the machine	[2]	15
72	W6 Array bounds violation: Overwrite any memory location with any desired value. Allows attacker to inject malicious code and take complete control of the machine	[2]	15
73	W7 Buffer overrun: Corrupt memory, crash the machine	[2]	15
74	W8 Buffer overrun, integer conversion bug: Corrupt memory until the machine crashes	[2]	15

75	W10 Buffer overrun: Overwrite return address on the stack. Allows attacker to inject malicious code and take complete control of the machine	[2]	15
76	W11 Array bounds violation: Information disclosure: read from potentially any memory address. Crash the machine	[2]	15
77	W12 Array bounds violation: Writes any 4-byte value to any address. Allows attacker to inject malicious code and take complete control of the machine	[2]	15
78	W13 Array bounds violation: Information disclosure: read a 4-byte value from any address	[2]	15
79	W14 Pointer arithmetic error: Crash machine. Could begin interpreting random memory locations as though they were .abo code	[2]	15
80	Three types of tokens used to potentially read and modify data in global memory. (no specifics given)	[3]	7
81	Error checking is inadequate for identifying and recovering from a failures in a damaged on disfunctional environment	[3]	11
82	Error handling makes it difficult to differentiate between malicious activity and failure	[3]	11
83	Error codes returned by the AV-OS system are ignored.	[3]	11
84	Attacker can hide preloaded votes (exact mechanism not given)	[5]	8
85	Vote counter allows integer overflows	[5]	8
86	Allows programming conditional behavior based on time, number of votes counted, and others	[5]	8
87	Enables AccuBasic program interaction over the LCD screen (can pose as normal firmware, for example)	[5]	20
88	AV-OS fails to check that the vote counters are zero at the start of election day	[2]	18
89	The code does contain a check to ensure that it will not accept more than 65535 ballots. It manipulates vote counters values without first checking them for overflow as 16-bit if more than 65535 votes are cast, the vote counters will wrap	[2]	18
90	Cards' integrity is protected by symmetric MAC rather than a much more ideal pk signature	[2]	20
91	Default cryptographic keys that are hard-coded into the source code	[2]	20
92	PIN is stored in an obfuscated format, but this obfuscation offers limited protection due to reliance on hard-coded magic constants	[2]	21
93	No requirements documents, architecture documents, design documents, threat model documentation, or security analysis documents	[2]	24
94	Modify the vote counters on the memory card to pre-load it with some non-zero number of votes for each candidate	[2]	25
95	Replace the AccuBasic script with a malicious script that falsely printed a zero report showing zeros, even though the vote counters were in fact not zero	[2]	25
96	Attacker can maliciously preload some of the vote counters with fraudulent non-zero values	[2]	27
97	Enables AccuBasic program interaction over the LCD screen (can pose as normal firmware, for example)	[2]	28
98	Several fields not covered by checksums	[2]	29

99	V1 Array bounds violation: Overwrite any memory address within ◆215 bytes of the global context structure with a 2-byte value that the adversary has partial control over. Might allow attacker to inject malicious code and take complete control of the machine	[2]	14
100	V2 Format string vulnerability: Crash the machine; read the contents of memory within a narrow range	[2]	14
101	V3 Input validation error: Choose any location on the memory card and begin executing it as .abo code; could be used to conceal malicious .abo code in unexpected locations, or to crash the machine	[2]	14
102	V4 Array bounds violation: Memory corruption; crash the machine	[2]	14
103	V5 Double-free() vulnerability: Overwrite any desired 4-byte memory address with any desired 4-byte value. Allows attacker to inject malicious code and take complete control of the machine	[2]	14
104	V6 Array bounds violation: Memory corruption: overwrite any memory address up to 216 bytes after the global context structure with a 2-byte value that the adversary has no control over. Might allow overwriting vote counters	[2]	14
105	V7 Buffer overrun: Memory corruption; crash the machine	[2]	14
106	V8 Buffer overrun: integer conversion bug: Memory corruption: overwrite up to 215 consecutive bytes of memory starting at global context structure. Might allow attacker to inject malicious code and take complete control of the machine. Might allow overwriting vote counters. Information disclosure: read any memory location 215 bytes away from global context structure. Crash the machine	[2]	14
107	V9 Buffer underrun: Memory corruption: overwrite up to 215 consecutive bytes of memory extending backwards from the global context structure. Might allow attacker to inject malicious code and take control of the machine. Might allow overwriting vote counters. Information disclosure: read any memory location within this window. Crash the machine	[2]	14
108	V10 Buffer overrun: Overwrite return address on the stack. Allows attacker to inject malicious code and take complete control of the machine	[2]	14
109	V11 Array bounds violation: Information disclosure: read from potentially any memory address. Crash the machine	[2]	14
110	V12 Array bounds violation: Write any 2-byte value to any address up to 216 bytes after the global context structure. Might allow attacker to inject malicious code and take complete control of the machine. Might allow overwriting vote counters	[2]	14
111	V13 Array bounds violation: Information disclosure: Read any 2-byte value from any address up to 216 bytes after the global context structure	[2]	14
112	V14 Pointer arithmetic error: Crash machine. Could begin interpreting random memory locations as though they were .abo code	[2]	14
113	V15 Unchecked string operation: Machine might crash or become unresponsive	[2]	14
114	V16 Unchecked string operation: Overwrite stack memory. Might allow attacker to inject malicious code and take complete control of the machine	[2]	14

115	Some default values not found in the user's guide	[3]	13
116	Default values defined in the source code	[3]	13
117	Contains "malign" undocumented functions	[3]	13
118	AccuBasic executable files (.abo) on the server are not authenticated	[5]	9
119	Failure to regularly install MS Windows updates	[11]	20
120	Server enables "autorun" feature, thus software can be installed via CD by anyone with physical access	[11]	20
121	Open USB port on the back of the server	[11]	21
122	CD is Bootable	[11]	21
123	Database password and audit logs stored within the database itself	[11]	21
124	Incomplete implementation of the SSL 3.0	[11]	21
125	Lack of routine server security practices, such as firewall protection, etc.	[11]	21
126	BIOS not password protected	[11]	22

Software Review and Security Analysis of the Diebold Voting Machine Software

Supplemental Report

David Gainey, Michael Gerke, and Alec Yasinsac

Security and Assurance in Information Technology Laboratory
Florida State University
Tallahassee, Florida

August 10, 2007

For the Florida Department of State

Software Review and Security Analysis of the Diebold Voting Machine Software

Supplemental Report

1 Introduction

On May 14th 2007, the Florida Department of State (FLDoS) commissioned an independent expert review of Diebold Voting System software version 1.96.8, by a team led by Florida State University's (FSU) Security and Assurance in Information Technology (SAIT) Laboratory [1]. The team issued its final report on July 27, 2007 [2] (hereafter referred to as the SAIT Diebold Report without citation). Shortly thereafter, the Florida Secretary of State (FLSoS) declined to certify the system that employed software version 1.96.8 as submitted and required Diebold to make substantive repairs [3] in order to continue the certification process.

On August 7th, FLDoS asked SAIT Laboratory to conduct a supplemental review of the Diebold software version 1.96.9 that the vendor submitted through the Independent Test Authority, in response to the FLSoS letter. The request's exclusive goal is to determine if the specific, required software repairs were accomplished. This report is the culmination of the requested supplemental review.

1.1 The Analysis' Scope

The scope of the investigation was exclusively to determine if the four required repairs listed on the first attachment page in [3] are evident and complete in version 1.96.9. The four flaws identified for required repairs are:

1. The Signature Flaw [3, par. 3.5]
2. Attacker Can Hide Preloaded Votes [3, par. 3.8.1.4]
3. AccuBasic Scripts Can Be Misused [3, par. 3.9.3] (noted as paragraph 3.9.1 in [3]).
4. Unchecked String Operation: Allows Overwrite of Stack Memory [3, paragraph 3.9.5]

1.2 Security Disclaimer

This report reflects the narrow investigative scope requested by FLDoS. These results are not comprehensive in any sense, nor is this report an endorsement of the system's overall security. We examined only a small subset of the flaws from the SAIT Diebold Report. All other flaws identified in that report remain in the code base, including vulnerability to a sleepover attack that may allow an intruder to manipulate vote computation or worse. Significant, critical vulnerability remains in this code base independent of repairs documented in this report.

Our exclusive purpose is to provide a technical assessment regarding four specific flaws. We defer interpretation and application of these technical results to FLDoS.

1.3 The Review Process

As a supplemental review, the basic terms of the SAIT Diebold Report also apply to this document, including all definitions, disclaimers, and findings that are not specifically addressed herein.

Upon receipt of the new code version from FLDoS, the team electronically compared the modified firmware version (1.96.9) to the version reviewed in the SAIT Diebold Report (1.96.8). Our review focused on the detected code differences, which included approximately two hundred lines of code, including comments. This small code volume facilitated quick response and increases confidence in our results.

With the code differences in hand, we reviewed each required change in [3] and compared it to the code differences to determine if they impacted a required change. Our findings below detail instances

where the changes impacted the required changes and further report our judgment regarding the technical impact that the accumulated changes have on the required flaws overall.

1.4 The Software Review Team

David Gainey is a computer science graduate student, a member of SAIT Laboratory, and is a member of the technical staff at the Florida State University Office of Technology Integration. He was a member of the original SAIT Diebold review team [2] and has evaluated a variety of voting system software products.

Michael Gerke is a computer science graduate student and a member of SAIT Laboratory at Florida State University. He is also presently employed by the Florida Department of State, Division of Elections. He was a member of the original SAIT Diebold review team [2] and has evaluated a variety of voting system software products.

Alec Yasinsac is an Associate Professor of Computer Science at Florida State University, a co-Director of SAIT Laboratory, and is the lead Principal Investigator on this project. He also led the SAIT teams that conducted the Florida Congressional District Thirteen voting system software review [4] and the Diebold review [2] that this project supplements.

1.5 Limits of this Study

1.5.1 Laboratory Results

There is always risk in conducting laboratory analysis for systems that operate outside the laboratory environment. At best, laboratory analysis presents a clinical perspective, often allowing scientists and investigators to isolate issues that cannot be economically evaluated during normal testing, operational testing, or during normal operations, but does not necessarily accurately capture more broad interactions.

Computer software has many properties that lend it favorably to laboratory analysis. For example, software is naturally created in a laboratory environment where programmers intentionally isolate specified functionality into their coding assignments. Subsequent laboratory testing and demonstration is often used to convince purchasers that the software meets its functional and foundational requirements. Thus, laboratory analysis is inherent, and well-understood, in the software process.

Some have questioned the applicability of laboratory analysis with regard to voting system software. However, there is no doubt that flaws identified in the SAIT Diebold Report exist and are technically exploitable, as is reflected in the serious response by the FLSoS [3]. Moreover, the SAIT Diebold Report discussed some common elections procedures and scientifically identified prerequisites and mitigations of this vulnerability where it exists.

Unfortunately, there is no clear existing procedure for testing voting system exploits in a non-laboratory setting. Because election integrity is sacrosanct, no reputable scientist or election official would engage a realistic security test in a real election. Worse yet, there is no way to reliably determine the likelihood that software vulnerability prerequisites are met even in the face of rigorous election procedures.

There are many systems that share this testing-limited property; these systems are sometimes termed “moon shot” systems. Fortunately, there is a coding approach that is suitable for critical, testing-limited systems; this is termed “high assurance” development. Since voting systems cannot be subjected to dangerous security analysis while in operation, they should be developed for high assurance. Until this occurs, elections officials face an unnecessarily high risk and they must exercise significantly expanded election security procedures to mitigate known and unknown software vulnerability.

1.5.2 Completeness Limitations

Any study involving systems and source code of the complexity of the Diebold Optical Scan system raises questions of completeness: could the investigators have missed problems? In this report, we document our efforts and assumptions to allow others to evaluate the thoroughness of our study.

Moreover, it is well-known that testing is an inherently incomplete process and that no testing process can guarantee absence of flaws. Regarding even the isolated flaws that we evaluated, we cannot guarantee that the changes or repairs that removed, mitigated, or reduced these flaws did not cause regression faults. While we did look for regression faults, it is possible that undetected regression faults may be more dangerous than the flaws that they fixed. Thus, we do not offer guarantees, but only our best professional and academic opinions.

Our conclusions were guided solely by the source code examined. We did not generate binaries for this source code to compare against a verified build. If this source does not precisely match that used in executing modules, our results may not apply.

2 Findings

We group our findings by the flaw that they describe. Where flaws overlap, we cross-reference to ensure consistency and to provide context to related flaws.

The Team's primary findings are that one of the four targeted flaws is fixed and the three other flaws are significantly improved in the revision. As an example of what we mean by "significantly improved", one of the two identified buffer overflow flaws was fixed, while the remaining buffer overflow flaw has no known exploit.

2.1 The Signature Flaw

As described in the SAIT Diebold Report,

> *... the hand-coded RSA signature verification is insecure and signatures generated with the implemented method can be forged.*

The original code applies a standard SHA1[1] hash to efficiently protect data integrity, then applies an RSA public key signature [5] across the hash value to guarantee its authenticity. This is a commonly accepted practice and, absent two subtle implementation miscues, it can provide strong authenticity assurances. The flaw occurred in the previous version because the RSA signature is longer than the SHA1 hash and the length difference was not properly handled[2], allowing an attacker to sign, or forge a signature on, an arbitrary message without knowing the signing key.

The crux of the problem was that when the signature was verified, only the SHA1 part of the signature was checked, allowing an astute attacker to leverage the unchecked bits to generate effective forgeries. The attack is facilitated because the implementation used another commonly accepted practice of selecting a key exponent of three (3), which was recently shown to expose RSA signatures to some subtle attacks.

In the revision, the vendor corrected the signature verification by modifying the code to verify the entire signature. They extended their repair by incorporating padding into the signature process in the commonly accepted way (appending the hash algorithm, etc.), so the process now controls and checks the entire signature content.

This algorithm fixes the signature flaw, thus preventing forgeries. We analyzed the code in some detail and it appears to be properly implemented.

---

[1] See FIPS PUB 180-1

[2] http://nvd.nist.gov/nvd.cfm?cvename=CVE-2006-4339

2.2 Attacker Can Hide Preloaded Votes

This original flaw, highlighted by the Hursti attack [6], involves improper counter management. Specifically, variables used to count votes were not properly protected and verified, allowing an attacker to preload votes by manipulating the counters on the memory card. Hursti's attack involved balancing incremented vote counters with other decremented (logically negative value) counters. In the modified code, the vendor implemented new counter routines that prevent counter faults.

The class of preloaded vote attacks identified under this flaw in the SAIT Diebold Report did not involve functional counter operation. Rather, these issues concern the fact that vote count fields on the memory card are not integrity protected. This allowed an attacker to edit fields to change votes with the only technological challenge being to appropriately set error detecting checksum values.

The vulnerability (as paraphrased from paragraph 3.8.1.4 of the SAIT Diebold Report) was:

1. Prepare a memory card with preloaded votes and insert it into the terminal after the zero report is printed. The modified count would be shown on the terminal's display.
2. Extract a memory card from the terminal after several votes had been registered, modify the counters to redistribute votes but also to maintain the same total vote count, and reinsert the memory card in the terminal if an attacker can gain suitable access to the terminal.
3. Prepared a memory card with a predetermined vote count and wait until the number of votes entered on the card were normally registered on the terminal. At that point, the attacker could replace the official memory card with the forged card and the terminal's display would reflect the correct, expected number of votes cast and the precinct count printout would show the injected votes.
4. Construct an interpreter script that disregards the counters on the memory card and prints the preloaded counts.

In the 1.96.9 revision, the subject ballot definition file, including the counter fields, is still not integrity protected. However, an attacker's ability to remove and reinsert a rogue memory card into the terminal is limited as a side-effect of the AccuBasic script signature repair (Section 2.1).

Under normal operations, card removal results in a properly reported system failure and forces a user initiated system shutdown. Since the script verification status is stored in volatile memory, card removal and reinsertion normally forces signature re-verification at the next script access, thus a rogue card would be detected when results were printed unless the card's script were properly signed.

Since the signature flaw repair now prevents signature forgery, this limits an attacker to: (1) Re-use a script from a properly signed, previously used memory card to create a rogue memory card or (2) Re-use the existing card after modifying the ballot definition file content, which requires a significantly longer attack-time terminal access period, or (3) Bypass the card removal shutdown process. We consider the first to be the most likely attack avenue and discuss the third further in Section 2.3 below.

We find that this flaw is significantly improved.

2.3 AccuBasic Scripts Can Be Misused

Unlike the ballot definition information on the removable memory cards, the AccuBasic script[3] is protected by the RSA/SHA1 signature. Thus, repairing the signature (see Section 2.1 above) reduces AccuBasic vulnerability. Signature verification now prevents an attacker from routinely inserting a rogue memory card with a custom script during terminal supervision gaps within the voting period.

However, due to the protocol applied to access the memory card, the vulnerability is not completely eliminated by the signature repair. The signature on the memory card script is only checked when the

[3] AccuBasic scripts are stored as ".abo" files in their source format. Since there is no file system on the memory cards, we refer to these programs as "scripts" when referring to them on the memory card.

script is first exercised, for example when the zero tape is printed. As noted in Section 2.2 above, under normal operations, card removal results in a properly reported system failure and a resulting user-initiated system shutdown. Since the script verification status is stored in volatile memory, card removal and reinsertion would normally force signature re-verification at the next script access. If an attacker could bypass the card removal shutdown process, they may be able to inject a rogue card with an unsigned script.

Clearly, script protection relies on subtle interactions between otherwise independent components, in this case, proper script verification depends on the card removal shutdown procedure always completing properly. This is a dangerous development practice that increases future regression fault likelihood.

We find that the noted technical flaw is significantly improved, in that while vulnerability still exists, we did not find an exploit.

2.4 Unchecked String Operation: Allows Overwrite of Stack Memory

As the SAIT Diebold Report indicated, while the vulnerable string operations for these two flaws still existed in the 1.96.8 code, the repairs made to correct other flaws mitigated the vulnerability these flaws introduced in versions prior to 1.96.8.

In the resubmitted version (1.96.9), one of these buffer overflow flaws is fixed, while the other is not. We are not aware of any exploit for the other, unfixed buffer overflow vulnerability.

3 Conclusions

As requested by the Florida Department of State, we analyzed the vendor's changes made in response to the FLSoS certification letter. We determined that the Signature Flaw repairs were accomplished and that the Preload Votes, Unchecked String Operations, and AccuBasic Script Misuse flaws were significantly improved.

We further note that the changes reflect a weak security approach. While these procedures appear to have fixed one of the required flaws and reduced vulnerability in the others, it is precisely these types of procedures that led to the original flaws. Presence of these procedures suggests that additional flaws will emerge in the application as it is now written and are even more likely to occur as regression faults in future versions.

We conclude by re-stating that this report does not constitute a comprehensive security analysis. We limited our investigation to four specific flaws. In spite of repairs made, significant security vulnerability continues to exist in the code base.

This system's overall security properties must be considered in terms of the complete elections environment to determine the practical impact of all remaining flaws.

4 References

1 "Software Review and Security Analysis for Diebold Voting Machine Software.", Joint Florida Department of State and Florida State University Statement of Work, May 14, 2007

2 Ryan Gardner, Alec Yasinsac, Matt Bishop, Tadayoshi, Kohno, Zachary Hartley, John Kerski, David Gainey, Ryan Walega, Evan Hollander, and Michael Gerke, "Software Review and Security Analysis of the Diebold Voting Machine Software", Final ReportFor the Florida Department of State, July 27, 2007, http://election.dos.state.fl.us/pdf/SAITreport.pdf

3 Kurt Browning, Florida Secretary of State Letter to Mr. David Byrd, Diebold Election Systems, dated July 31, 2007, http://election.dos.state.fl.us/pdf/SAITbrowningLetter.pdf

4 A. Yasinsac, D. Wagner, M. Bishop, T. Baker, B. de Medeiros, G. Tyson, M. Shamos, and M. Burmester, “Software Review and Security Analysis of the ES&S iVotronic 8.0.1.2 Voting Machine Firmware, Final Report”, Security and Assurance in Information Technology Laboratory, Florida State University, February 23, 2007, http://election.dos.state.fl.us/pdf/FinalAudRepSAIT.pdf.

5 R.L Rivest, A. Shamir, L. M. Adleman, "A Method for Obtaining Digital Signatures and Public Key Cryptosystems", CACM, Vol. 21, No. 2, Feb 1978, pp. 120-126

6 The Black Box Report, SECURITY ALERT: Critical Security Issues with Diebold Optical Scan Design, July 4, 2005.

Security Assessment of the Diebold Optical Scan Voting Terminal

A. Kiayias L. Michel A. Russell A. A. Shvartsman

UConn VoTeR Center and
Department of Computer Science and Engineering,
University of Connecticut
{akiayias,ldm,acr,aas}@cse.uconn.edu

with the assistance of
M. Korman, A. See, N. Shashidhar, D. Walluck

October 30, 2006

Abstract

We present an independent security evaluation of the AccuVote Optical Scan voting terminal (AV-OS). We identify a number of new vulnerabilities of this system which, if exploited maliciously, can invalidate the results of an election process utilizing the terminal. Furthermore, based on our findings an AV-OS can be compromised with off-the-shelf equipment in a matter of minutes even if the machine has its removable memory card sealed in place. The basic attack can be applied to effect a variety of results, including entirely neutralizing one candidate so that their votes are not counted, swapping the votes of two candidates, or biasing the results by shifting some votes from one candidate to another. Such vote tabulation corruptions can lay dormant until the election day, thus avoiding detection through pre-election tests.

Based on these findings, we describe new safe-use recommendations for the AV-OS terminal. Specifically, we recommend installation of tamper-resistant seals for (i) removable memory cards, (ii) serial port, (iii) telephone jacks, as well as (iv) screws that allow access into the terminal's interior; failure to seal *any single one* of these components renders the terminal susceptible to the attack outlined above. An alternative is to seal the entire Optical Scan system (sans ballot box) into a tamper-resistant container at all times other than preparation for election and deployment in an election. An unbroken chain of custody must be enforced at all times. Post-election audits are also strongly advised.

The Diebold AccuVote Optical Scan voting terminals described in this report are going to be used in November 2006 election in several precincts in the State of Connecticut. The terminals are provided by the LHS Associates of Massachusetts. VoTeR Center personnel assisted the Office of the Connecticut Secretary of the State in developing safe use procedures for the Optical Scan terminals for this election. The procedures in place for the election includes strict physical custody policy, tamper-resistant protection of the equipment, and random post-election audits.

1 Introduction

The subject of this paper is the **AccuVote Optical Scan** voting terminal (AV-OS) manufactured by Diebold, Incorporated, Election Systems division.

Figure 1: The AccuVote Optical Scan voting terminal (AV-OS). The terminal is shown prior to it being locked to the ballot box, with its front panel visible and showing two control buttons (lower left corner) and the memory card slot with the card sealed in (lower right corner).

An important benefit of using the optical scan technology in electronic voting systems is that it naturally yields a voter-verified paper trail—the actual "bubble sheet" ballots marked by the voters. This differentiates optical scan electronic voting from DRE (direct recording electronic) electronic voting terminals (such as the Diebold AccuVote TS and TSx terminals) that provide a digital interface for voting during the elections. We note that the current generation of the DRE terminals—especially paperless ones—have received substantial criticism due to a number of critical security vulnerabilities, such as those reported in [1, 2, 7]. Even when a DRE terminal is equipped with a printer, the computer-generated paper trail cannot be directly considered voter-verified, and it is possible for a faulty DRE to print spontaneous ballots while unobserved. Further development of the DRE technology is necessary for it to become a trustworthy alternative.

While optical scan voting is freed from some of the perils of paperless trails or computer generated paper trails, the election still relies on the terminal to electronically add the votes and report the results; this introduces the possibility of attacks that interfere with these basic tabulation and reporting tasks. Such an attack against the AV-OS was demonstrated recently in [3]. This attack was particularly devastating as it initialized the counters of the terminal to negative or positive vote counts while still forcing the machine to report a valid zero-count initialization. This can lead to biased election results and corrupted election counts. The operation of the AV-OS system is in part governed by the instructions stored in a *memory card* that is inserted into the terminal for the duration of the election. The attack of [3] employed a memory card reader/writer to modify the card prior to election and bring it to an *invalid initial state*. When a maliciously altered card is used in an election, it records biased results that are successfully tabulated by the terminal.

Given that the attack in [3] required tampering with the memory card directly, one way to mitigate the attack is to somehow ensure that the memory card stays in place sealed into the terminal throughout the period that the machine is in use or is in transit to and from the polling places. Alternatively (and most effectively) one could employ a cryptographic integrity check, however this would require modifications to the firmware

of the system (presumably by the manufacturer). A second way to mitigate the attack would be to execute a pre-election test, hand-count the ballots, and compare this to the report of the terminal.

> Given the facts summarized above, the pressing question is whether the security measures of (1) sealing the memory card into the terminal, and (2) performing pre-election testing with hand-counted ballots, are sufficient to prevent an attack against an election employing the AV-OS.
>
> Our findings answer this question in the negative.

In particular we show that *even if the memory card is sealed and pre-election testing is performed*, one can carry out a devastating array of attacks against an election *using only off-the-shelf equipment* and *without having ever to access the card physically or opening the* AV-OS *system box*. Our attacks include the following:

1. **Neutralizing candidates.** The votes cast for a candidate are not recorded.
2. **Swapping candidates.** The votes cast for two candidates are swapped.
3. **Biased Reporting.** The votes are counted correctly by the terminal, but they are reported incorrectly using conditionally-triggered biases.

Our attacks exploit the serial communication capability of the AV-OS and demonstrate how the attacker can easily take control of the machine and force it to compromise its sealed-in resident memory card. Moreover, we demonstrate how one can make the AV-OS appear to be uncompromised to an evaluator that performs a pre-election test by voting hand-counted ballots, or to an evaluator that examines the audit reports that are produced by the terminal. A corrupted terminal will in fact appear to be faithfully reporting any election procedure that is conducted prior to the day of the election, only to misreport its results on the day of the election.

We also present a low-tech "digital ballot stuffing" attack that is made possible due to the mechanical characteristics of the optical scan reader. This simple attack enables any voter to vote an arbitrary number of times using two Post-it® notes. This attack makes it imperative to have the terminal under constant supervision during elections.

The vulnerability assessment provided in this paper is based only on experimentation with the system. At no point in time had we used, or had access to, internal documentation from the manufacturer or the vendor, including internal machine specifications, source code of the machine's operating system, layout of the data on the memory card, or the GEMS ballot design and tabulation software. We developed attacks and software that compromises the elections from first principles, by observing system's behavior and interaction with its environment. Based on this fact, we conclude that attackers with access to the components of the AV-OS system can reverse-engineer it in ways that critically compromise its security, discover the vulnerabilities presented herein and develop the attacks that exploit them.

2 Basic Characteristics of the System

Our findings are based on the evaluation of an AV-OS system that was delivered to us by LHS Associates of Methuen, MA as a part of an evaluation on behalf of the State of Connecticut. The AV-OS election system consists of two components: the AccuVote Optical Scan voting terminal (the AV-OS terminal) and the ballot design and central tabulation system (GEMS, for Global Election Management System). These components have the following characteristics:

- The GEMS software is installed on a conventional laptop PC and includes a ballot design system and a tabulation system.
- The specifications of an election are downloaded onto a 40-pin 128KB Epson memory card present in the AV-OS. It should be noted that the memory card has been discontinued by Epson, and no reader/writer for this type of medium is readily available in the market.
- The AV-OS systems provided to us contained the firmware version 1.96.6. It is equipped with an optical scanner, a paper-tape dot-matrix printer, a LCD display, a serial communication port, and telephone jacks leading to a built-in modem. For election deployment the system is secured within a ballot box so that no sensitive controls or connectors are exposed to the voter.

3 Security Vulnerabilities

We briefly describe the new vulnerabilities that were discovered during our evaluation process. A detailed presentation of these vulnerabilities is available in an extended version of the report that can be provided on a need-to-know basis.

The AV-OS leaks the memory card contents: The AV-OS terminal allows any operator to obtain a dump of its installed memory card contents without any authentication control. In particular, given access to an AV-OS machine one can obtain all the information that is stored in the memory card in a matter of seconds. In order to obtain this information, it is sufficient to use an off-the-shelf RS-232 serial cable (null modem cable) and a laptop. The AV-OS performs no authentication test to ensure that a trusted system is present on the other side while the dump is delivered in cleartext form. Moreover, the terminal does not prompt the operator for a password in order to produce such memory dump. It is easy to identify the election data when observing a memory dump; other sensitive information, including the *password (PIN) and audit records* associated with the memory card can also be reconstructed from the dump. Alternatively, the same dump can be obtained by using the built-in modem on the AV-OS to transmit the data to a remote PC.

The communication between AV-OS and GEMS is unauthenticated: During the initialization of a machine for election the GEMS system communicates with the AV-OS terminal to write the initial election setup to the memory card. No encryption or cryptographic authentication is performed during this transmission. The serial line protocol does use a cyclic redundancy check (CRC) mechanism for error control. While the CRC polynomial used is standard, the details of the protocol are undocumented by the manufacturer; as such, this is a de facto lightweight authentication mechanism. However, it is possible to reverse-engineer the whole protocol, including the CRC scheme formula (as we have done in our assessment). The lack of cryptographic authentication opens the possibility for an unauthorized attacker computer to impersonate the GEMS system to the terminal (this is one of the ingredients of our main election compromising attack in the next section).

Executable code within the AV-OS memory card: Each memory card contains executable code that is used for printing the reports. The code is written in a proprietary symbolic language. Such executable files are identified as `.abo` (AccuBasic Object) bytecode. The possibility to modify the code that prints the results opens the possibility to corrupt machines and coerce them into misinterpreting their counters. The presence of conditionals and arithmetic in the language enables bytecode "malware" to operate even conditionally on the state of the machine and thus appear to operate properly in some occasions

while misreporting the results in others. While this vulnerability was already known from [3, 7] it was not employed as a tool to conditionally misrepresent the outcome of an election (but rather as a tool to hide a corrupted initial election setup).

Multiple feeding: The sensor that detects that a ballot has been fully inserted into the optical scan slot is positioned in the right hand side of the feeder. This opens the possibility for multiple feeding if voters are left unattended during the time that they insert their ballot into the terminal (cf. §4.2).

4 The Attacks

We now present new attacks against the AV-OS system that use the vulnerabilities described above. The first attack entirely compromises the election process assuming that the attacker has a few minutes of access to the AV-OS terminal prior to election time. The second attack shows how voters can vote multiple times using the same ballot if they are left unattended to use the terminal during an election.

4.1 Compromising the Election

By compromising the election we refer to an attacker's capability to put the AV-OS in a state where it miscounts the ballots that are inserted into the machine. For example an election would be compromised if the votes received by two candidates are swapped or if the votes of a candidate were nullified.

To streamline our attack, we have developed a proof-of-concept software package that processes card dump data, extracting the ballot layout, password (PIN), and audit information, and computes a serial payload to reprogram the card. We emphasize that our software was developed by observing the AV-OS system during normal operation, without access to any technical information about the system, its internals, or access to the source code of AV-OS or GEMS. Specifically, the attack was developed with precisely the same information and access to the system that is normally available to, for example, election administrators (poll workers and other town officials).

Equipped with a laptop and a regular RS-232 null modem serial cable, an attacker needs only to gain physical access to an AV-OS terminal prior to the election. Furthermore, the attacker needs no knowledge of the particulars of the election he is to undermine (such as exact candidates' names, ballot layout, precinct names, or any kind of passwords). The whole process can be completed in a matter of a few minutes. In the following we perform a step-by-step demonstration of the attack.

Step 1 : Gaining physical access[1]**.** In Figure 2 the AV-OS terminal is shown locked within the ballot box. This would (presumably) be the state that the terminal is found prior to the election. At this stage, the system has been initialized with all the election data and its removable memory card is sealed with a tamper evident seal.

The first thing an attacker must do is gain access to the front side of the AV-OS that is concealed by the ballot-box. If the box is unlocked or the attacker has the keys this is straightforward. We note that the locks used are regular pin tumbler locks similar to those found in filing cabinets, office drawers or other standard computerized equipment. If the attacker lacks the key, picking the lock can be done in a short amount of time ranging from seconds to minutes (it is feasible even for someone who has never done it before using information available online, e.g., [6]). Picking the lock requires no special equipment: in fact two standard paper clips are sufficient, see Figure 3.

[1] If the attacker has access to the election-ready voting terminal prior to its being locked within the ballot box, proceed to Step 2.

Figure 2: The AV-OS terminal installed within the ballot box and locked.

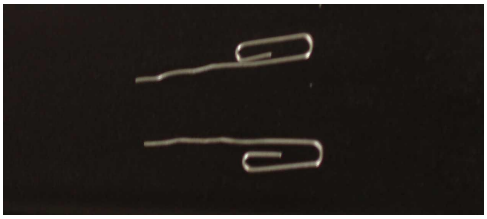


Figure 3: The two clips an attacker can use to pick the AV-OS ballot box front lock and gain access to the machine.

Once the lock is opened, the front side of the machine is freed and the attacker can access the Yes/No buttons located on the left of the front panel of the AV-OS, see Figure 4.

Step 2 : Dumping the memory card contents. Once the AV-OS terminal is accessible from the front, the attacker can pull it slightly outwards and obtain access to its back side. There, a number of standard connection ports are available including a RS-232 serial port and a telephone line jack. The attacker uses a standard serial cable to connect the machine to her laptop, see Figure 5.

In order to prepare for the attack the laptop must capture the data sent to its serial port by the AV-OS. Though we have written software which automates this portion of the attack, a standard terminal emulator would suffice.

Once the serial cable is in place, the attacker turns on the machine using the on/off switch located on the right of the machine's back panel while simultaneously depressing the two buttons on the front panel. This results in the AV-OS entering in a special diagnostic mode. The terminal asks for no password or other identification from the operator in order to enter into such a mode. One of the options that is available to the attacker in this mode is to dump the contents of the installed memory card through the serial line. This is the option that the attacker selects and the AV-OS dumps the card contents, see Figure 5.

It takes roughly two minutes to receive (and parse) the card dump that the AV-OS transmits. The dump of the card is sent in cleartext and the only component that is hidden is the PIN that enables the attacker to

Figure 4: The front side of the AV-OS with the Yes/No buttons accessible.

enter into a special "supervisor mode." This is the mode that poll-workers have access to during election time. The 4-digit PIN is contained in an obfuscated form at a fixed location in the dump. As part of assessment, we reverse-engineered the method used to obfuscate the PIN. The election compromising software de-obfuscates and prints the supervisor mode access PIN, see Figure 6. In addition, our software decodes the "audit history" that appears in the card dump, including the entry containing the initialization timestamp for the card as well as any other entry in the transaction log of the terminal.

In the same screenshot our main menu is presented that has the following options: (1) neutralize candidate votes, (2) swap candidate votes, (C) print candidate list, (D) display election info, (Q) quit and send data. Using the options (C) and (D) the attacker can obtain all information about the election including the ballot layout.

Step 3 : Ballot design remapping. In order to understand the specifics of the attack, we overview the election setup of the AV-OS system. Each candidate and race has a unique identifier. The candidates for each race are encoded together with an (x,y) coordinate (cf. Figure 6), which corresponds to the bubble on the paper ballot sheet that the voters mark in order to vote for that particular candidate. The ballots are printed taking into account this configuration. The correct correspondence between printed ballots and internals of the memory card is essential for the election to go through uncompromised. This correspondence is one of the aspects of the election system that the attack subverts.

To neutralize a candidate in a specific race, the attacker simply maps the (x,y) coordinate of the candidate to some location that is beyond the ones used for the election (note that most coordinates are in fact unused; thus it is trivial for the software to recover such a position). In the current implementation of the election compromising software the location selected for neutralizing a candidate whose coordinates are (x,y) is $(x-1,y+1)$. The $(x-1,y+1)$ pair is suitable as this choice will not affect the value of the *checksum* that the terminal computes from the ballot layout data.

An equally devastating attack is swapping two candidate's votes. Following the previous rationale if the bubble coordinates assigned to candidate A are (x_A,y_A) and the bubble coordinates assigned to candidate B are (x_B,y_B), by simply swapping the coordinates one effectively makes AV-OS count a vote for candidate A

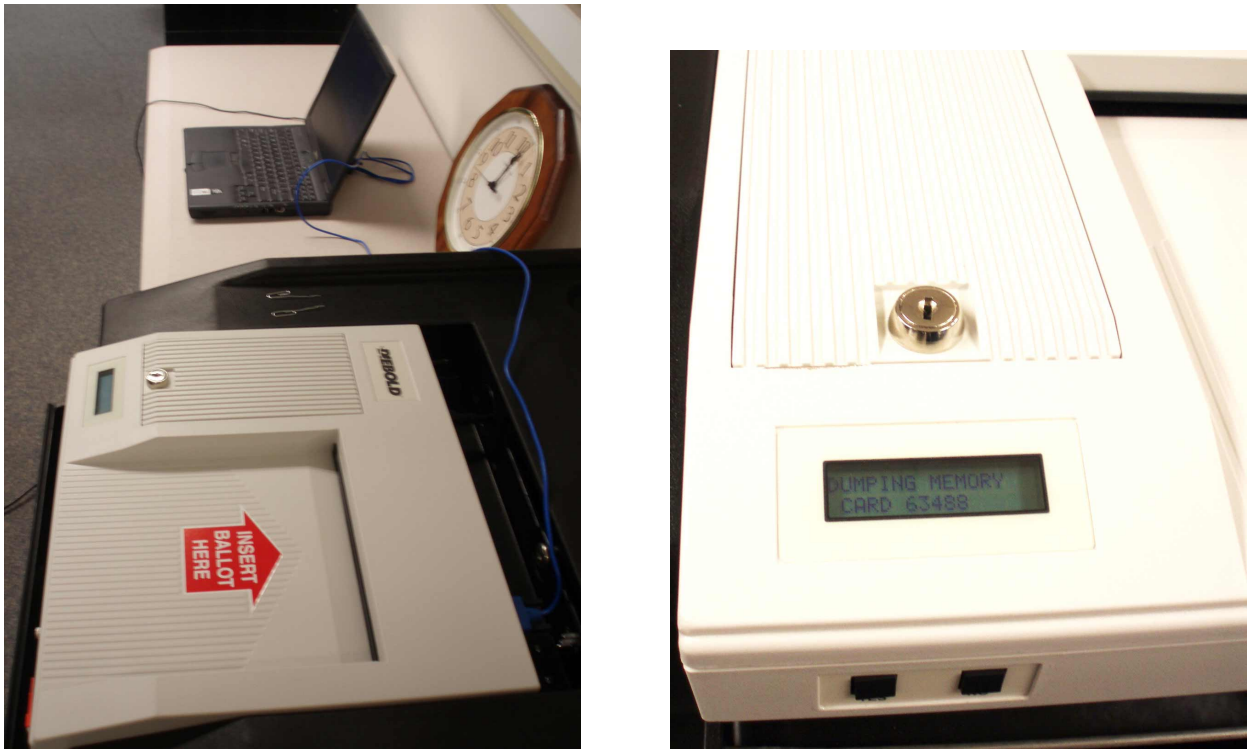


Figure 5: (*Left*): The setup for compromising the AV-OS. A standard laptop is connected through its serial port to the serial port of the AV-OS machine. (*Right*): Dumping the memory contents through the serial line. Anyone with physical access to the AV-OS can perform this operation since this function does not require any authentication. The number shown in the LCD screen is the amount of remaining bytes before the dump is completed (each card holds 128Kb of data).

as a vote for candidate *B* and vice versa.

These modifications are built-in into the election compromising software that also includes additional payloads for biasing the reporting functionality of the terminal. What needs to be performed next by the attacker is to use the AV-OS to reprogram the memory card with this altered election data.

Step 4: Adjusting the AV-OS clock to agree with the card's initialization timestamp. When the election compromising software processes the dump of the memory card, it also recovers the time and date at which the card was originally programmed for the election. To insure that this timestamp is preserved in the audit history of the new image of the card to be created in Step 6 below, the attacker would need to reset the clock of the AV-OS so that it agrees with the recovered timestamp. The option to (re)set the clock appears in Diagnostic Mode, obtained by restarting the machine with both buttons pressed.

Step 5: Temporarily disabling the AV-OS printer. When the AV-OS terminal is initialized it prints a tag that can be used for auditing the system and contains the date and time of the initialization as well as some other control information. Given that the attacker will reinitialize the system, in order to prevent the AV-OS from printing such tag, the attacker must disable the printing functionality by selecting the corresponding choice available in the "supervisor menu" of the terminal that is accessible by using the de-obfuscated PIN. This step is optional as the attacker may simply discard the printout, nevertheless the fact that the attacker can disable

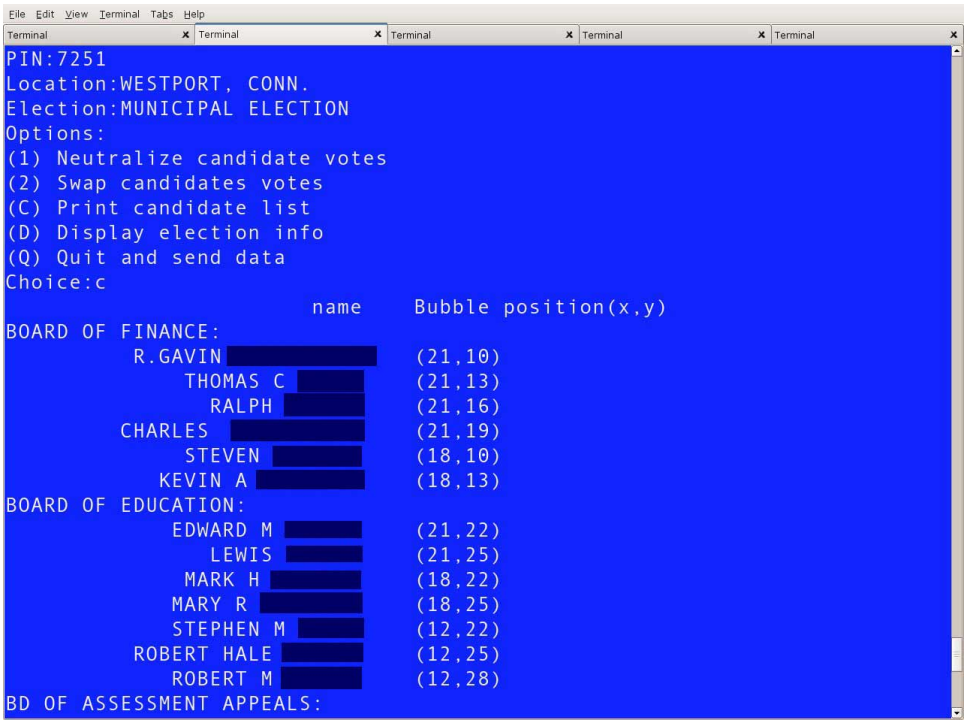


Figure 6: (*Top part*) : The main menu of the election compromising software. The de-obfuscated PIN is prominently presented. (*Bottom part*) : The listing of candidates for some of the races and the corresponding bubble sheet coordinates for each candidate. (This snapshot is touched-up to black out the last names used in this fictitious race.)

the printer makes the attack more stealthy.

Step 6 : Impersonating the GEMS system. Once the AV-OS clock is reset and the printer is shut-off the attacker sets the AV-OS terminal in supervisor mode.

In supervisor mode, AV-OS can format the contents of the memory card and accept communication from the GEMS system to initialize the election. The attacker takes advantage of the fact that the AV-OS does not use any strong cryptographic identification check to authenticate the sending entity and hence it can impersonate the GEMS system.

Using the election compromising software the attacker prepares a forged election payload. The preparation of this payload is based on the reverse engineering of the communication between the GEMS system and AV-OS that we performed as part of our vulnerability assessment. The software prepares a fake communication transcript that appears to be originating from GEMS. The transcript contains the election details recovered from the memory dump together with a number of malicious alterations such as candidate swaps, candidate neutralizations and corrupted bytecode reporting functionality.

In this particular run of the attack we made the following choices (see Figure 6):

- The votes of “Thomas C” and “Kevin A” in the Board of Finance race are swapped.
- The candidate “Mark H” in the Board of Education race is neutralized.

Figure 7: (*Left*) : The attacker enters the PIN to enter supervisor mode. (*Right*) : The AV-OS is requesting communication from the GEMS system to overwrite the memory card contents with the forged election setup.

In Figure 7 we show how the attacker enters the 4-digit PIN that was recovered from the memory dump to gain access to the options of the supervisor mode of the terminal. In order to start the machine in supervisor mode the unit needs to be turned off and restarted while simultaneously depressing the 'Yes' button. Subsequently the attacker chooses to erase the memory card contents, and the card is formatted. Once the contents of the memory card are erased the unit would request to be initialized from the GEMS system. In Figure 7 the AV-OS terminal requests communication from the GEMS system. The attacker furnishes to the terminal the forged communication transcript.

Step 7 : Completing the attack. Once the forged communication is transmitted through the serial port the compromise of the terminal has been successfully completed. The attacker will reset the clock to the current time using the diagnostic mode and will activate the printer.

After this step, the AV-OS terminal will be found by poll-workers in its expected pre-election state. The terminal will appear to be functioning normally for all operations during the election. Interestingly, the terminal is not even in an invalid initial state after a card has been compromised in the way described above. The defect is found only in the mapping between the candidates and the bubbles in the printed ballot sheets that has been rearranged in malicious intent.

The total time required to compromise the card is only a few minutes, depending on the dexterity of the attacker in picking the lock of the ballot box. If the attacker possesses the key or minimum lock-picking expertise the locks can be picked in about 30 seconds and the whole attack can be carried out at a leisurely pace in less than 5 minutes.

4.1.1 Test Election

We have conducted an election with the compromised AV-OS terminal to illustrate the attack's effectiveness. We have prepared 10 ballots as shown in Figure 11 in the appendix where we selected 7 votes for candidate "Thomas C" and 5 votes for candidate "Kevin A" in the race for the "Board of Finance." We also voted thrice for candidate "Mark H." in the race for the "Board of Education." As shown in the report presented in Figure 12, the results of candidates "Kevin A." and "Thomas C" are reversed with "Kevin A" receiving 7 votes and "Thomas C" receiving 5 votes. On the other hand the "Mark H." appears to have received no votes at all.

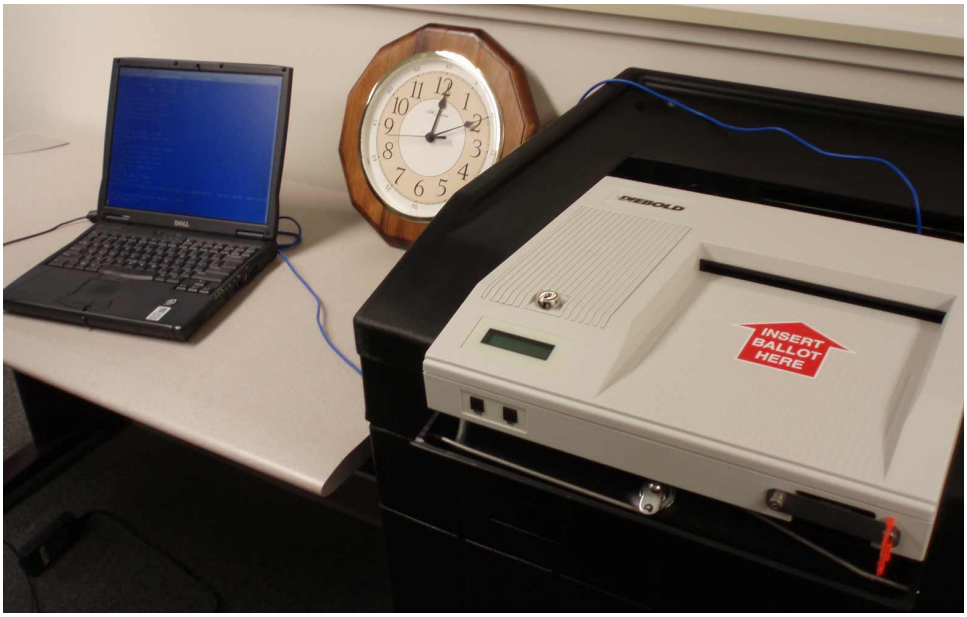


Figure 8: The AV-OS terminal has been compromised. The election specifications that are stored in the memory card contain an incorrect mapping of candidates to bubble locations, replacing the mapping necessary to conduct a proper election as configured on the printed ballots.

4.1.2 Identifying a Compromised Machine

The election compromising software takes special measures to conceal the tampering of the terminal and make the detection of a compromised machine difficult even when the election officials follow the recommended procedures to test the integrity of the system. In particular, the election compromising software embeds the following functionality into a corrupted machine: a compromised AV-OS terminal records the actual results improperly but *when the poll-workers execute a test election* prior to opening the polls it still prints out the results correctly.

This double deceitfulness of a compromised machine — to behave improperly in the real election but behave properly when tested — can be achieved as follows: the report functionality of the terminal is altered by the election compromising software so that it corrects its misaligned counters in the event that the ballot count is too low (which would correspond to the case when the poll officials test a small batch of hand-counted votes) or when the date and time is prior to the real election time.

In other words, in standard computer security terminology, the attacker can plant a "time-bomb" in the terminal. Before the election, the program in the terminal's card inverts the swapped counters to conceal the malicious behavior (the swapping of votes). When the time of the election comes, the illicit behavior is triggered automatically. This sensitivity to time will prevent poll-workers that perform the standard test procedures from revealing that a machine is compromised prior to the election.

4.1.3 Compromised Election Results

An election is deemed corrupted when the miscounted results get tabulated into the overall election totals. If this is performed manually using the printed receipts that are produced by a corrupted terminal, the corruption

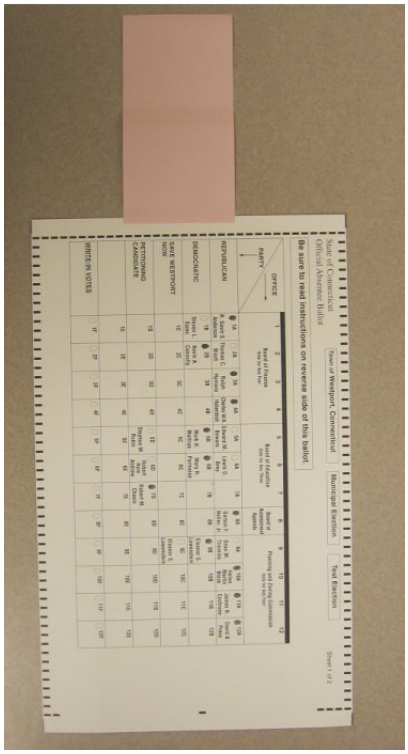


Figure 9: The prepared ballot used for the re-voting attack. Ballot stuffing is as easy as obtaining a couple of standard Post-It notes if the terminal is not closely monitored during an election.

of an election would be immediate. The results can also be tabulated electronically, by consolidating memory cards using a terminal and communicating such results to the central tabulation system implemented in GEMS. The compromised cards that contained the improperly aligned counters are accepted by the central tabulation system without any warning or any other indication that they may be corrupted.

4.2 Multiple Voting Using Two Post-It® Notes

In this section we present a simple low-tech attack that is based on the following facts regarding the ballot feeding mechanism of the AV-OS terminal:

- The ballot-feed sensor is located on the right side of the slot. Feeding paper into the left side does not trigger the feed mechanism.
- Once a ballot is fed into the AV-OS, the rollers cease. It is thus possible to retract a ballot from the other side of the rollers. This is easily done even when the AV-OS has been properly locked into position atop the ballot box. Moreover, this can be done very quickly, so that the amount of extra votes is only limited to the amount of time the voter is able to spend alone with the ballot box on election day.
- The machine is unable to recognize ballots that have already been cast. Although the AV-OS verifies an election identifier which is global to every ballot in a precinct, it allows the same ballot to be cast as many times as desired.

We demonstrate how this vulnerability can be very easily exploited by any voter during the actual election if she is allowed to operate the machine without being observed by a poll-worker. See Figure 9 for an example of an AV-OS ballot with the two Post-it notes affixed to its side. The attacker in this case is allowed to use the machine while unattended and he can pull out and re-insert the shown ballot so that the same vote is cast multiple times.

5 Recommendations for Safe Use

Given that the AV-OS terminal is merely a "bubble sheet" counting device and not a DRE system, there is no fear that the actual votes will be lost (since they are preserved in the voter generated paper trail). Nevertheless, the fact that the votes are not lost does not necessarily imply that they will be counted correctly. The attacks presented herein suggest that the AV-OS system has serious security defects in its design that demand strict observance of safe use guidelines. Based on our findings we propose the following:

- It is important to seal in a tamper evident fashion not only the memory card slot but also the serial port and the phone jacks of the terminal. Instead of sealing these sockets it is possible to disconnect them internally from the motherboard so that they are disabled as shown in Figure 10, although this approach has the disadvantage that its implementation cannot be verified without opening the system box.

  Protecting the device with tamper-evident seals to secure it against opening of the system box is equally important. Opening the system box of the device not only makes the memory card exposed but also enables one to circumvent sealed serial ports by directly connecting a properly configured cable to the motherboard.

  A complete approach involves protecting the entire AV-OS device within a tamper-evident enclosure at all times other than the actual deployment in an election. (This approach is being taken in Connecticut where the system carrying case is secured by a tamper-resistant numbered seal, with the seal number checked at all transit points.)

- Chain of custody should be strictly observed, from the point of initialization of the terminal to the time it returns to long-term storage after an election. The procedures for transporting and handling the equipment must be defined in advance (such procedures are in place in Connecticut).

- The memory card should never be allowed to be outside the AV-OS terminal (in fact this is the approach taken in the State of Connecticut). Given that no cryptographic integrity check is employed by the AV-OS memory card management, the moment the card is removed from a terminal it can be considered to be compromised. Unfortunately even if the memory card is sealed in the terminal, as our attack demonstrated in §4.1, the system does not guarantee an uncompromised election unless the remaining ports and case of the terminal are sealed in a tamper-evident fashion as well. While not directly addressed in this report, it is also necessary to safeguard the firmware chip in the AV-OS system. The chip contains the AccuBasic interpreter, and it is designed to be replaceable due to version changes (such as the change from 1.94.$*$ to 1.96.$*$). It is imperative the ensure that the right chip is installed prior to the AV-OS machines being deployed for election.

- With proper precautions, the re-voting attack (cf. §4.2) should not be possible. The ballot should be filled out in private in an area separate from the ballot box, but inserted in public, while preserving the secrecy of the vote (several states, including Connecticut already use this approach). The AV-OS ballot

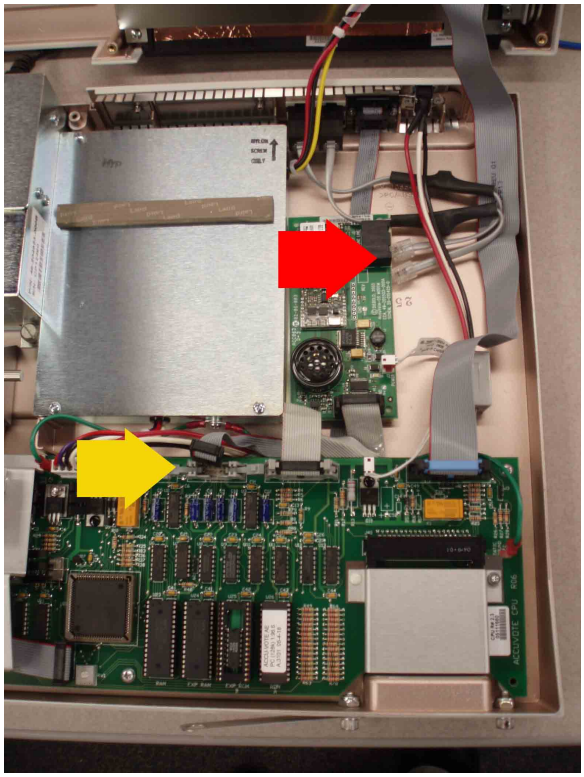


Figure 10: Disabling the serial line and modem of a GEMS terminal. The upper arrow (red in color layout) shows the disconnected telephone cables from the internal modem, whereas the lower arrow (yellow in color layout) shows the disconnected serial cable from the motherboard.

box should be kept under close supervision by poll workers during elections. Poll-workers must not be allowed to take their eyes off the machine, and should be wary of attempts at distraction.

- Finally, a post-election random audits involving hand counting of the ballots are highly recommended. (Such random audits will be conducted in the State of Connecticut.)

Acknowledgments. The authors thank the Office of the Connecticut Secretary of the State for supporting the production of this report.

References

[1] Tadayoshi Kohno, Adam Stubblefield, Aviel D. Rubin and Dan S. Wallach, Analysis of an Electronic Voting System, IEEE Symposium on Security and Privacy 2004, IEEE Computer Society Press, May 2004.

[2] Ariel J. Feldman, J. Alex Halderman, and Edward W. Felten, Security Analysis of the Diebold AccuVote-TS Voting Machine, September 13, 2006
`http://itpolicy.princeton.edu/voting`

[3] Harri Hursti, Critical Security Issues with Diebold Optical Scan Design, Black Box Voting Project, July 4, 2005
`http://www.blackboxvoting.org/BBVreport.pdf`

[4] Harri Hursti, Diebold TSx Evaluation, Black Box Voting Project, May 11, 2006
`http://www.blackboxvoting.org/BBVtsxstudy.pdf`

[5] Susan Pynchon, The Harri Hursti Hack and its Importance to our Nation, Florida Fair Elections Coalition, January 21, 2006. `http://www.votetrustusa.org`

[6] Theodore T. Tool, MIT Guide to Lock Picking, 1991
`http://people.csail.mit.edu/custo/MITLockGuide.pdf`

[7] David Wagner, David Jefferson and Matt Bishop, Security Analysis of the Diebold AccuBasic Interpreter, Voting Systems Technology Assessment Advisory Board, University of California, Berkeley, February 14, 2006.

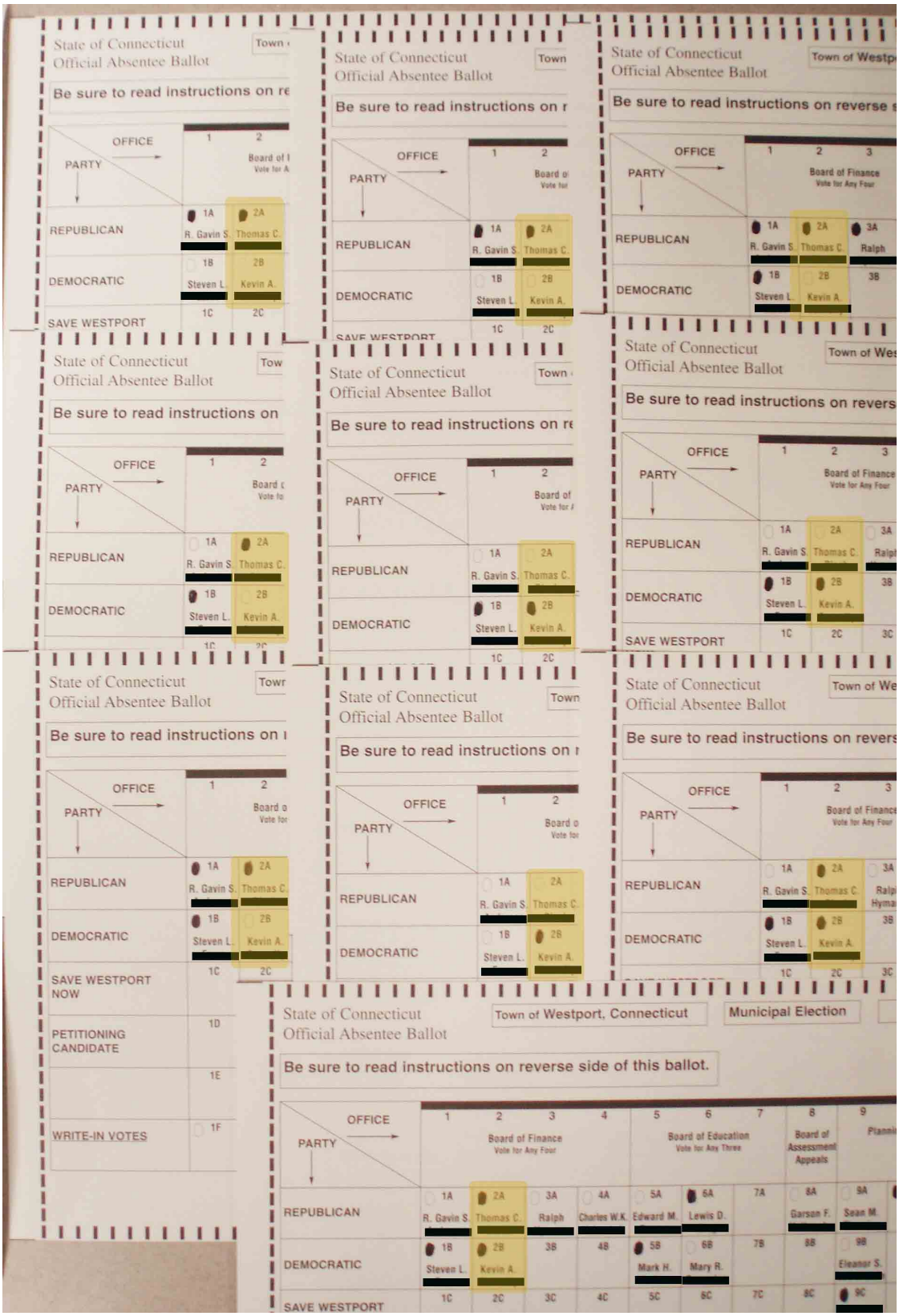


Figure 11: The collection of the ballots used for the test election on the compromised AV-OS. Candidate "Thomas C." receives 7 votes and candidate "Kevin A." receives 5 votes.

Figure 12: The outcome of the election as reported by the compromised AV-OS terminal. The votes of candidates "Thomas C." and "Kevin A." has been swaped. Candidate "Mathias H." appears to have received no votes.

VoTeR Center
UConn Voting Technology Research Center
PI : A. Shvartsman, Ph.D.
Co-PIs: A. Kiayias Ph.D., L. Michel Ph.D., A. Russell Ph.D.

AccuVote Optical Scan Vulnerabilities and Safe Use

October 27, 2007

Version 0.1

Introduction

The development of modern computerized voting technology, while empowering voters, also introduces vulnerabilities due to the possibility of accidental or malicious interference with the voting processes. Recent reports have identified numerous such vulnerabilities. While it is difficult to provide absolute guarantees that proper operation of a particular voting terminal cannot be interfered with, in certain cases – once the vulnerabilities of the terminal is carefully assessed – it may be possible to design policies and procedures to be followed by the election workers so as to enable the safe use of the terminal by the voters and to ensure that the election results are correctly recorded.

The subject of this document is the AccuVote Optical Scan voting terminal (AV-OS). We begin by overviewing optical scan technology, contrasting it with the touch screen technology, and discussing the vulnerabilities issues of AV-OS terminals. We then explicitly enumerate the vulnerabilities identified for the AV-OS terminals and the impact of these vulnerabilities with respect to the possibility of malicious tampering with the vote counting and the election results. We conclude with the discussion of threat mitigation and recommendations for safe use of AV-OS terminals. The recommendations include imposing a strict chain-of-custody policy on AV-OS terminals and memory cards, pre-election testing of memory cards, and post-election audits.

Optical Scan Technology in Perspective

An important benefit of using the optical scan technology in electronic voting systems is that it naturally yields a voter-verified paper trail — the actual “bubble sheet” ballots

marked by the voters. This differentiates optical scan electronic voting from DRE (direct recording electronic) electronic voting terminals (such as the Premier's AccuVote TS and TSx terminals) that provide a digital interface for voting during the elections. We note that the current generation of the DRE terminals — especially paperless ones — have received substantial criticism due to a number of critical security vulnerabilities, such as those reported in [KSRW04, Hursti06, Princeton06, Berkeley06, UConn07]. Even when a DRE terminal is equipped with a printer, the computer-generated paper trail cannot be directly considered voter-verified, and it is possible for a faulty DRE to print spontaneous ballots while unobserved. Further development of the DRE technology is necessary for it to become a trustworthy alternative.

While optical scan voting is free from some of the perils of paperless trails or computer generated paper trails, the election still relies on the terminal to electronically add the votes and report the results; this introduces the possibility of attacks that interfere with these basic tabulation and reporting tasks. Such an attack against the AV-OS was demonstrated by Hursti [Hursti05]. This attack was particularly devastating as it initialized the counters of the terminal to negative or positive vote counts while still forcing the machine to report a valid zero-count initialization. This can lead to biased election results and corrupted election counts. The operation of the AV-OS system is in part governed by the instructions stored in a memory card that is inserted into the terminal for the duration of the election. The attack of [Hursti05] employed a memory card reader/writer to modify the card prior to election and bring it to an invalid initial state. When a maliciously altered card is used in an election, it records biased results that are successfully tabulated by the terminal. Given that the attack in [Hursti05] required tampering with the memory card directly, one way to mitigate the attack is to somehow ensure that the memory card stays in place sealed into the terminal throughout the period that the machine is in use or is in transit to and from the polling places. Alternatively (and most effectively) one could employ a cryptographic integrity check, however this would require modifications to the firmware of the system (presumably by the manufacturer). A second way to mitigate the attack would be to execute a pre-election test, hand-count the ballots, and compare this to the report of the terminal.

Given the facts summarized above, the pressing question was whether the security measures of (1) sealing the memory card into the terminal, and (2) performing pre-election testing with hand counted ballots, are sufficient to prevent an attack against an election employing the AV-OS.

Our own findings [UConn06] answer this question in the negative. In particular we showed that even if the memory card is sealed and pre-election testing is performed, one can carry out a devastating array of attacks against an election using only off-the-shelf equipment and without having ever to access the card physically or opening the AV-OS system box. Examples of our attacks include the following:

1. Neutralizing candidates. The votes cast for a candidate are not recorded.
2. Swapping candidates. The votes cast for two candidates are swapped.
3. Biased Reporting. The votes are counted correctly by the terminal, but they are reported incorrectly using conditionally-triggered biases.

Our attacks exploit the serial communication capability of the AV-OS and demonstrate how the attacker can easily take control of the machine and force it to compromise its sealed-in resident memory card. Moreover, we demonstrate how one can make the AV-OS appear to be uncompromised to an evaluator who performs a pre-election test by voting hand-counted ballots, or to an evaluator who examines the audit reports that are produced by the terminal. A corrupted terminal will in fact appear to be faithfully reporting any election procedure that is conducted prior to the day of the election, only to misreport its results on the day of the election.

The vulnerabilities documented in [UConn06] were developed by experimentation with the system. At no point in time had we used, or had access to, internal documentation from the manufacturer or the vendor, including internal machine specifications, source code of the machine's operating system, layout of the data on the memory card, or the source of the GEMS ballot design and tabulation software. We developed attacks and software that compromises the elections from first principles, by observing system's behavior and interaction with its environment. Based on this fact, we conclude that attackers with access to the components of the AV-OS system can reverse-engineer it in ways that critically compromise its security, discover the vulnerabilities presented herein and develop the attacks that exploit them.

Additional vulnerabilities are documented (the "Berkeley Report") by the members of the California Voting Technology Assessment Advisory Board (VSTAAB) with the assistance of the University of California, Berkeley, and issued on February 14, 2006 [Berkeley06]. The report documents a number of vulnerabilities due to the AccuBasic interpreter of the AV-OS and TSx voting terminals. What was discovered is that the implementation of the interpreter left the doors open for the election results to be tampered with by using some of the standard "hacking" techniques such as buffer overruns and array bound violations. These attacks can be devastating by leading to rogue code completely taking over the AV-OS system.

The vulnerabilities documented in the "Berkeley Report" complement both the "Hursti Attack" [Hursti05] and our findings [UConn06]. The "Berkeley Report" underscores the need for strict chain of custody of AV-OS, memory cards, and GEMS systems, and strong policies for who and how is to access these systems and devices. The report also lists several short- and longer-term mitigation strategies, all of which are clearly sensible, and should be implemented. Subsequent comprehensive reports in California [CA07] and Florida [FL07] confirm and catalog earlier findings.

Summary of Vulnerabilities

We now explicitly enumerate the vulnerabilities that exist in the AV-OS voting terminal as used in the State of Connecticut.

1. The AV-OS "leaks" the memory card contents: The AV-OS terminal allows any operator to obtain a dump of its installed memory card contents without any authentication control. In particular, given access to an AV-OS machine one can obtain all the information that is stored in the memory card in a matter of seconds. In order to obtain this information, it is sufficient to use an off-the-shelf RS-232 serial cable (null modem cable) and a laptop.
2. The AV-OS performs no authentication test to ensure that a trusted system is present on the other side while the dump is delivered in cleartext form. Moreover, the terminal does not prompt the operator for a password in order to produce such memory dump. It is easy to identify the election data when observing a memory dump; other sensitive information, including the password (PIN) and audit records associated with the memory card can also be reconstructed from the dump. Alternatively, the same dump can be obtained by using the built-in modem on the AV-OS to transmit the data to a remote PC.
3. The communication between AV-OS and GEMS is unauthenticated: During the initialization of a machine for election the GEMS system communicates with the AV-OS terminal to write the initial election setup to the memory card.
4. No encryption or cryptographic authentication is performed during this transmission. The serial line protocol does use a cyclic redundancy check (CRC) mechanism for error control. While the CRC polynomial used is standard, the details of the protocol are undocumented by the manufacturer; as such, this is a de facto lightweight authentication mechanism. However, it is possible to reverse-engineer the whole protocol, including the CRC scheme formula (as we have done in our assessment). The lack of cryptographic authentication opens the possibility for an unauthorized attacker computer to impersonate the GEMS system to the terminal.
5. Executable code within the AV-OS memory card: Each memory card contains executable code that is used for printing the reports. The code is written in a proprietary symbolic language. Such executable files are identified as .abo (AccuBasic Object) bytecode. The possibility to modify the code that prints the results opens the possibility to corrupt machines and coerce them into misinterpreting their counters. The presence of conditionals and arithmetic in the language enables bytecode "malware" to operate even conditionally on the state of the machine and thus appear to operate properly in some occasions while misreporting the results in others.
6. The AccuVote interpreter (residing in the firmware of the AV-OS terminal) is open to being corrupted by maliciously constructed .abo bytecode. This enables an

attacker to deliver malicious code through the memory card, resulting in arbitrary behavior of AV-OS during its deployment in an election.

7. AV-OS system does not check that valid firmware is contained in the PROM (read-only memory) chip. This in essence allows an attacker to load their own code in the AV-OS terminal that can result in arbitrary system behavior. This is a "hardware" attack that is more difficult to execute, but its consequence is identical to substituting a maliciously designed AV-OS look-alike terminal for a real terminal. (It is this "capability" that allows us to turn AV-OS into a memory card reader to speed-up dumping of the card contents for testing purposes.)

Impact of the Vulnerabilities

As the consequence of the vulnerabilities described above, AV-OS systems can be tampered with under the following circumstances.

1. Memory cards are accessed by unauthorized personnel after they are programmed and before they are inserted into AV-OS terminals.
2. AV-OS terminals are accessed by unauthorized personnel (with or without memory cards inserted).
3. AV-OS terminals are accessed by unauthorized personnel after memory cards inserted before an election deployment (whether or not the AV-OS have been tested with the inserted cards).

Additional Considerations

The printing of physical ballots is currently done as a separate process, only indirectly related to the programming of the GEMS database and subsequent loading of the election data into the memory cards. It is important to verify that the printed ballots indeed correspond to the election data contained in the memory cards. It is possible and advisable to construct test decks of ballots that explicitly check for correctness of printed ballots.

Finally, as we pointed out, the loading of the memory card from the GEMS system includes an executable program that is stored in the GEMS environment. It is also important to check that this program correctly deals with vote tabulation and printing, and that it does not include any extraneous, erroneous, or malicious code.

Threat Mitigation and Recommendations for Safe Use

Vulnerabilities such as the ones described above suggest the possibility of tampering with elections that can be exploited by a sophisticated attacker. Still, in many cases it can be possible to devise specific procedural and technological measures to successfully thwart

the attacks even without requiring any modification to a terminal. Thus, once a comprehensive listing of attack vectors against a voting terminal has been completed, it is possible to develop mitigation methodologies that may enable the safe use of a voting terminal in an election procedure despite the existence of vulnerabilities.

Given that the AV-OS terminal is merely a “bubble sheet” counting device and not a DRE system, there is no fear that the actual record of the decisions made by voters will be lost (since they are preserved in the voter generated paper trail). Nevertheless, the fact that the votes are not lost does not necessarily imply that they will be counted correctly. The attacks presented herein suggest that the AV-OS system has security defects in its design that demand strict observance of safe use guidelines. Based on our findings we propose the following.

1. It is important to seal in a tamper evident fashion not only the memory card slot but also the serial port and the phone jacks of the terminal. Instead of sealing these sockets it is possible to disconnect them internally from the motherboard so that they are disabled, although this approach has the disadvantage that its implementation cannot be verified without opening the system box.
2. Protecting the device with tamper-evident seals to secure it against opening of the system box is equally important. Opening the system box of the device not only makes the memory card exposed but also enables one to circumvent sealed serial ports by directly connecting a properly configured cable to the motherboard. Additionally this allows an attacker to replace the firmware chip. A complete approach involves protecting the entire AV-OS device within a tamper-evident enclosure at all times other than the actual deployment in an election. (This approach is being taken in Connecticut where the system carrying case is secured by a tamper-resistant numbered seal, with the seal number checked at all transit points.)
3. Chain of custody should be strictly observed, from the point of initialization of the terminal to the time it returns to long-term storage after an election. The procedures for transporting and handling the equipment must be defined in advance (such procedures are in place in Connecticut).
4. The memory card(s) must be stored in tamper-evident containers whenever it is outside the AV-OS terminal. Given that no cryptographic integrity check is employed by the AV-OS memory card management, the moment the election-ready card is stored in the open, or is removed from a terminal by unauthorized personnel, it can be considered to be compromised. Unfortunately even if the memory card is sealed in the terminal, as our attack demonstrated, the system does not guarantee an uncompromised election unless the remaining ports and case of the terminal are sealed in a tamper-evident fashion as well.
5. It is also necessary to safeguard the firmware chip in the AV-OS system. The chip contains the AccuBasic interpreter, and it is designed to be replaceable due to

version changes. It is imperative to ensure that the right chip is installed prior to the AV-OS machines being deployed for election.

6. Given that programmed memory cards are shipped to Connecticut from LHS Associates, it is important to verify that the received cards are indeed programmed correctly and that the received cards are indeed the cards that have been programmed per election data in GEMS system. This can be done by examining the memory cards upon their arrival in Connecticut. At least a random audit of the cards should be performed before the election.
7. Finally, a post-election random audits involving hand counting of the ballots are highly recommended. (Such random audits will be conducted in the State of Connecticut.) It is also advisable to perform post-election audits of the memory cards used in the elections.

Summary of Recommendations

The following procedures, once implemented, will substantially help ensuring the integrity of the use of AV-OS voting terminals in the elections in Connecticut.

- Implement strict chain-of-custody policy for AV-OS terminals.
- Implement strict chain-of-custody policy for memory cards.
- Implement pre-election testing of programmed memory cards.
- Implement post-election audits.

Additionally, it is important to ensure that the printed ballots correctly correspond to the election data programmed into the memory cards.

References

[Berkeley06] David Wagner, David Jefferson and Matt Bishop, Security Analysis of the Diebold AccuBasic Interpreter, Voting Systems Technology Assessment Advisory Board, University of California, Berkeley, February 14, 2006.

[CA07] California “Top-to-Bottom Review”, University of California, Berkeley, July 2007. http://www.sos.ca.gov/elections/elections_vsr.htm

[FL07] Florida “Software Review and Security Analysis of Diebold Voting Machine Software”, Florida State University, July 2007.

[Hursti05] Harri Hursti, Critical Security Issues with Diebold Optical Scan Design, Black Box Voting Project, July 4, 2005 http://www.blackboxvoting.org/BBVreport.pdf

[Hursti06] Harri Hursti, Diebold TSx Evaluation, Black Box Voting Project, May 11, 2006 http://www.blackboxvoting.org/BBVtsxstudy.pdf

[KSRW04] Tadayoshi Kohno, Adam Stubblefield, Aviel D. Rubin and Dan S. Wallach, Analysis of an Electronic Voting System, IEEE Symposium on Security and Privacy 2004, IEEE Computer Society Press, May 2004.

[Princeton06] Ariel J. Feldman, J. Alex Halderman, and Edward W. Felten, Security Analysis of the Diebold AccuVote-TS Voting Machine, September 13, 2006 http://itpolicy.princeton.edu/voting

[UConn06] A. Kiayias, L. Michel, A. Russell, and A. A. Shvartsman. Security Assessment of the Diebold Optical Scan Voting Terminal, UConn Voting Technology Research Center, October 30, 2006, available at http://voter.engr.uconn.edu/voter/Reports.html.

[UConn07] A. Kiayias, L. Michel, A. Russell, and A. A. Shvartsman. Integrity Vulnerabilities in the Diebold TSX Voting Terminal. UConn Voting Technology Research Center, July 16, 2007, available at http://voter.engr.uconn.edu/voter/Reports.html.

Integrity Vulnerabilities in the Diebold TSX Voting Terminal

A. Kiayias L. Michel A. Russell A. A. Shvartsman

UConn VoTeR Center and
Department of Computer Science and Engineering,
University of Connecticut
{akiayias,ldm,acr,aas}@cse.uconn.edu

with the assistance of S. Davtyan, A. See, N. Shashidhar

July 16, 2007

Abstract

This report presents certain integrity vulnerabilities in the Diebold AV-TSx Voting Terminal[1]. We present two attacks based on these vulnerabilities: one attack swaps the votes of two candidates and another erases the name of one candidate from the slate. These attacks do not require the modification of the operating system of the voting terminal (as it was the case in a number of previous attacks). These attacks against the voting terminal can be launched in a matter of minutes and require only a computer with the capability to mount a PCMCIA card file system (a default capability in current operating systems).

The security problems are present in the system despite the fact that a cryptographic integrity check appears to be employed in the voting system's memory card. The attacks presented in this report were discovered through direct experimentation with the voting terminal and without access to any internal documentation or the source code from the manufacturer.

1 Introduction

Direct Recording Electronic (DRE) refers to voting terminals that use an interface to enable a voter to record his vote directly in digital format. The tallying is performed internally by the terminal that maintains counters for each candidate and race. DRE terminals have been criticized for lack of verifiability that they perform the tallying appropriately. As a result many DRE terminals today employ a VVPAT (Voter Verified Paper Audit Trail) system: the terminal is equipped with a printer that produces a record reflecting the choices of the voter; the voter is supposed to verify the VVPAT record. After the election it is possible to perform a manual count using the VVPAT records.

[1]Note: The AV-TSx voting terminals are quite different from the AccuVote Optical Scan terminals, and the vulnerabilities presented in this report do not apply to the Optical Scan terminals used by the State of Connecticut. The AV-TSx terminals are not used in Connecticut.

The Diebold TSx voting terminal was recently criticized in [5] and [6] due the following discovered security flaws: (i) it was possible to relatively easily circumvent the bootstrapping process and modify the operational environment of the system; this was identified in [5] where it was found that no cryptographic checks were performed during bootstrapping. (ii) the key management was by default using a fixed hard-coded key (that was leaked in the Internet); this was identified in [6] where the importance of choosing fresh signing keys was highlighted.

Fixing the above problems would require changes in the boot-loading process as well as adherence to an appropriate key management practice to be followed by election officials. In [6], it was reported that the AV-TSx has the advantage compared to other terminals such as the optical scan voting terminal (AV-OS) also from Diebold that it uses a cryptographic integrity check to make sure that the contents of the card have not been tampered.

Our Results. In our investigation we verify that there appears to be cryptographic integrity checking in the AV-TSx memory card. Nevertheless, we discover that the scope of the integrity checking is not as "wide" as it should have been. In particular, we find that in certain files that control the layout of the slate, the integrity checking is performed at the file level but not at the slate placement level. This flaw in the scope of the integrity check enabled us to modify the slate layout without triggering any alert from the terminal. Moreover, we found that when contents of slate components were invalidated the terminal did not issue an alert but instead it chose to simply suppress the corrupted file.

Based on the above vulnerabilities we design and test two attacks against the AV-TSx terminal. In the first, the attacker wishes to swap votes received by two candidates. The attacker can be successful in performing such attack assuming that the sizes of the two files that define the candidate representation in the digital slate are of the same size. We found that is not a rare occurrence and in fact our test election contained such pairs of candidates. The swapping was applied to the name definitions of the two candidates and included the integrity check. In the second attack, the attacker simply wishes to make one of the candidates disappear from the slate. By modifying the file that defines the layout of the name of the candidate this is achieved by the system.

The terminal we used in the above experiments is shown in Figure 1 and is manufactured by Diebold, Incorporated, Election Systems division. All our findings were based on reverse engineering and at no moment we had access to internal information about the terminal or access to source code.

Given the above findings, the employment of AV-TSx in an actual election becomes problematic. This is the case as the modification of a card can be done with merely a PC that is PCMCIA capable. If this terminal is used in an actual election it is extremely important to keep the memory card sealed in place. Moreover, it is very important to modify the operating system so that the integrity check is extended in all the card contents.

We also note that our terminal appear to lack the exact bootstrapping vulnerabilities that were reported in [5] (but lacking access to any internal information / system source code we are not able to vouch if the bootstrapping is any more secure now).

2 Security Vulnerabilities

This section discusses several security vulnerabilities in the AV-TSx. The attacks in section 3 focus mainly on the vulnerabilities with respect to the memory card (section 2.2), though the other issues mentioned here should be taken into consideration.

2.1 Basic Characteristics of the System

The system used in this study included the following components:

Figure 1: The AccuVote TSx voting terminal.

AccuVote TSx voting terminal:

- Firmware version 4.6.4
- Bootloader version BLR7-1.2.1
- Windows CE Operating system version WCER7-410.2.1

GEMS software version 1.18 install on a laptop.

Ethernet cable to connect the two above.

The GEMS software is used to manage the ballot information, load the election data onto the AV-TSx, and to receive the results after the election.

2.2 Memory Card

2.2.1 Description

The memory card is a standard PCMCIA flash card with a FAT file system. The card contents include the following file hierarchy

/ (root directory)
  Election Data/
    N.xtr
    N.edb
    M.adt
    K.brs
  Trashcan/

where N, M, and K are 32 character strings consisting of 0-9 and a-f (i.e., a 128 bit hex number). The .xtr file contains the election data information, the .edb file stores database information, the .adt file is the audit log, and the .brs file is the ballot box. The election data file is a bundle that contains many Rich Text Format (RTF) files for the displayed candidate names, wave files for use in audio voting, images displayed during the voting process and information about the precinct. All these files are packaged together in a single .xtr file along with 128 bit integrity checks for each of them. Votes are encrypted using 128 bit AES [8] and placed in the .brs file.

2.2.2 Vulnerabilities

Election Data and Database File Each candidate name (in an RTF file) is packaged with a 128 bit integrity check, however, these are not used correctly. A failed integrity check should render a voting machine unresponsive. However, in our terminal, a failed check of an RTF file simply makes that file not appear on the screen, effectively removing that candidate as an option.

The candidate names that are printed for the voter verified paper trail are based on the same RTF file that is displayed to the voter. However, the name printed for the final results is based on data from the .edb file. Because of this, voters could be unaware of any discrepancies between their cast votes and the internally recorded votes. Such a problem can only be detected by performing a manual count of the ballots from the VVPAT and comparing with the printed final counts.

There is also no global check to ensure the entire election data is correct. For example, the RTF files for candidates could be swapped (as we demonstrate below), along with their integrity check. A proper global integrity check should catch such manipulation.

Ballot Box There appears to be no global global cryptographic signature of the card contents to verify the contents were not tampered with outside the machine. Without this, it may be possible to stuff the ballot box by creating a custom ballot box file. This may depend on insider information to obtain the correct AES key and ballot format, but could be a threat nonetheless. Any changes to the memory card outside the voting terminal should result in an error.

Upgrade Files and Backdoors As documented in [4], previous versions of the TS machine were susceptible to attacks through back door files. If present on the memory card, the machine would give the user full access to the OS, for debugging purposes. For TSx machine it was documented in [5] that the back door files, with the different filenames, still exist. These files, if present on the memory card, will start to be processed based simply on their filenames; in this way an attacker can tamper with the boot loading process. We remark that the bootstrapping process in our TSx machine may still function as it is impossible to conclude positively that they are not working without having access to properly structured upgrade files. Still we tested the filenames that worked for previous versions and they no longer seem to function; moreover we have been unable to discover any similar backdoors as yet. However, without looking at the source code of the software being run on the machine it is impossible to say that there is no such back door still present in the system. A similar threat exists for the upgrading mechanism. In previous versions, only the name of the upgrade file was used to identify a valid software upgrade located on the memory card. Naturally, this represents a grave security vulnerability if no proper integrity checks are being used to authenticate the software upgrade. During our test we did not have examples of legitimate upgrade files, so we have been unable to test whether this vulnerability remains in the current version.

2.3 Internal Storage

The AV-TSx hardware includes internal flash memory in which it stores ballot information and voting results. This is used, for example, to accumulate results from several voting machines each with their own memory card. Each card is inserted in turn and an accumulator function in the ballot station software reads in the stored results (i.e., some accumulator values are maliciously preinitialized).

Vulnerabilities The accumulation functionality requires inserting each memory card into a AV-TSx terminal so that the results can be merged with those already stored on the internal flash memory. However, without source code, it is not clear precisely how the AV-TSx determines the data to be merged. In particular, it is unclear whether or not a AV-TSx terminal could ship with a set of election results already present which could be merged with valid results.

2.4 Limited Auditing Ability

The auditing features of the AV-TSx are limited to election results and system modifications, including turning on/off, loading data, and changing settings. However, there is no (documented) way in which to examine the software currently installed on the machine. Ideally one could (with proper access) dump the contents of the internal storage containing the operating system and voting software (or a hash of these contents) in order to verify the machine was not tampered with. In light of the possible lack of a secure method of loading new software, as mentioned above, this auditing ability maybe useful.

3 The Attacks

The attacks we present were developed with precisely the same information and access to the system that is normally available to, for example, election administrators (supervisors, poll workers and other town officials). Note that to carry out the attack, one only needs physical access to the voting machine, without the privileges of an election administrator. An attacker only needs a few minutes with the card and a hex editor to perform the attack. In addition, an attacker may need to open the lock which covers the removable card. Furthermore, the attacker needs no knowledge of the particulars of the election he is to undermine (such as exact candidates' names, ballot layout, precinct names, or any kind of passwords). What the attacker needs is to find two .rtf strings which have the same size (first 4 bytes of the .rtf string contain the .rtf file size) within the .xtr file. The whole process can be completed in a matter of a few minutes. In the following we perform a step-by-step demonstration of the attack.

3.1 Gaining physical access

Prior to the election the terminal is presumably locked within the ballot box. The first thing an attacker must do is to gain access to the memory card of the AV-TSx machine that is concealed by the ballot-box. If the box is unlocked or the attacker has the keys this is straightforward. The fact that DIEBOLD appears to be using the same keys across machines makes it easier to unlock the ballot-box (we had two terminals and they both shared the same keys). From what we have found online, in contrast to the keys used by DIEBOLD for OS machines, the keys for AV-TSx machines are very difficult to copy because they do not use a standard size. However, a copy of this key is sent to every precinct as part of the supply kit. Additionally the keys assigned to each location are not individually numbered, nor is there any record of which key is assigned to each precinct [11].

3.2 Reading the memory card contents

Once the PCMCIA card is accessible the attacker can have an immediate access to its contents through the PCMCIA card reader. Note, that a lot of laptops nowadays are equipped with a PCMCIA card reader.

3.3 The details

In order to understand the specifics of the attack, the following gives an overview of the election setup of the AV-TSx system. The removable memory card with the election loaded on it contains four different types of files: BRS, ADT, EDB and XTR. The .xtr file contains the ballot data. The .edb file, which has the same name as the .xtr file, contains the encrypted data from the Election Data Base. The .brs file stores the votes. The .adt file stores the Machine Log. For the attacks we present we are particularly interested in the .xtr file contents. This file bundles .rtf, .wav, and .bmp files. The .rtf files include the candidate names and voting instructions. The .wav files include the candidate names spoken. There are two .bmp files that illustrate the insertion of the voting ID into the machine. Each file is stored by the following way:

4 bytes - filesize(N); N bytes - data; 16 bytes - checksum.

What the attacker needs to do is to find two candidates for which the .rtf file sizes are the same and swap these two .rtf files with their corresponding checksums. Note, if the checksums are not swapped, thus the data do not correspond to the checksum, the voting software will simply not display this file. There are two possible scenarios for carrying out the attacks. First is just to nullify the candidate name, and the second is to swap two candidates.

Nullifying Attack: As it was discussed earlier each .rtf file consists of 4 bytes file size, following the N bytes data, and 16 bytes checksum. If the checksum is not consistent with the data it will result in nullifying the candidate name. Thus, if a single bit is changed in the data part of the .rtf file, without altering the length of the file, then the corresponding .rtf file will simply not be displayed on the screen without causing any error while loading the election. For example, we attempted to alter a candidate's .rtf file by replacing a 'C' with a 'D' The corresponding cell is left blank, but the checkbox remains. Voting proceeds as usual. When printing the ballot, if there were no votes for the (now blank) candidate, then it is printed with no name for that candidate. For example, if we would originally have

....

[X] THOMAS C. XXXXXX

....

We now have

....

[X]

....

An example of the original untampered slate is given in the Figure 2.

The exact same slate after a candidate has been nullified is given in Figure 3.

When the election is finalized, the results are printed using the candidate's original name. This means that the name is in fact stored in two places:

1. A label in a database record
2. Within the formated .rtf file

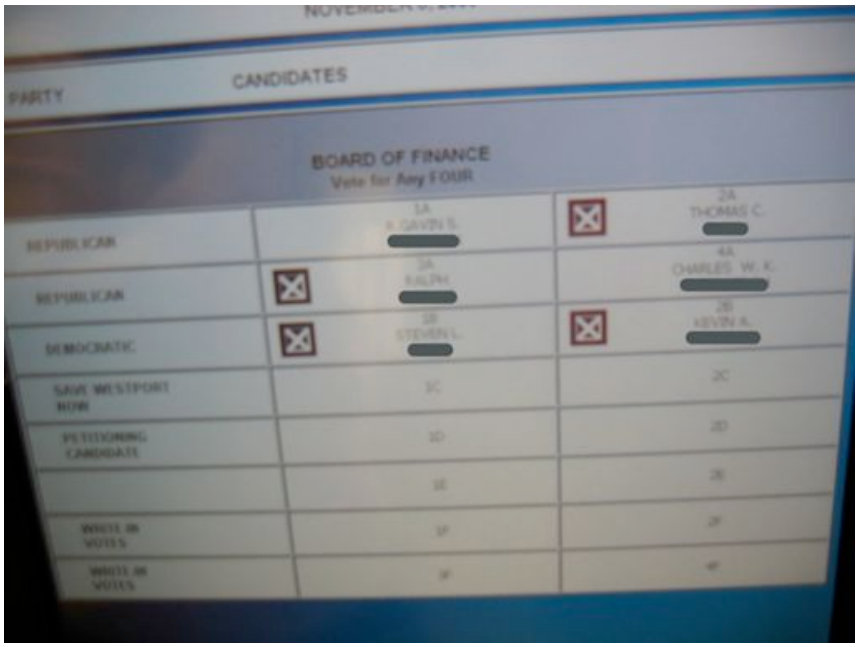


Figure 2: The original, not untampered, slate. Some choices have been made by the voter.

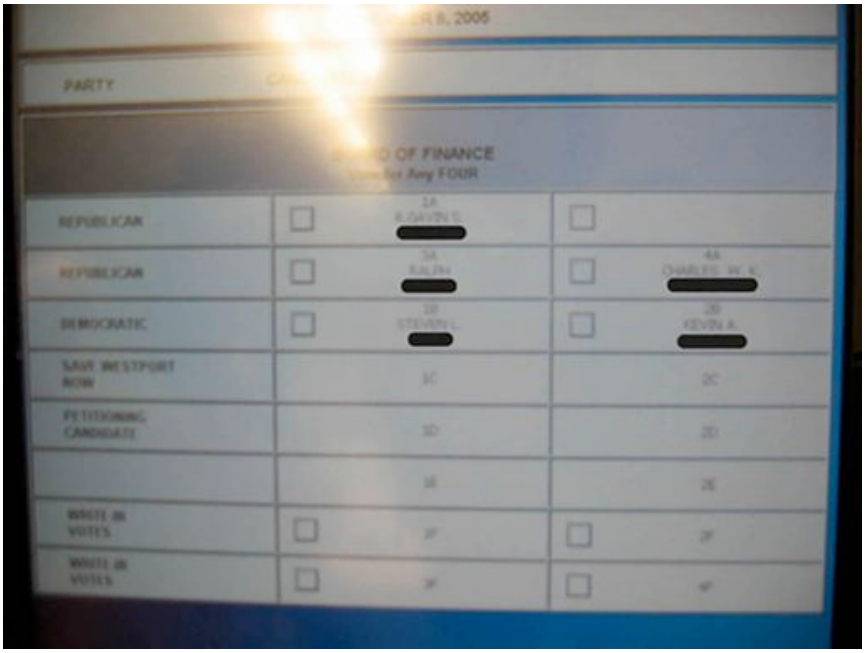


Figure 3: The slate with the nullified candidate name.

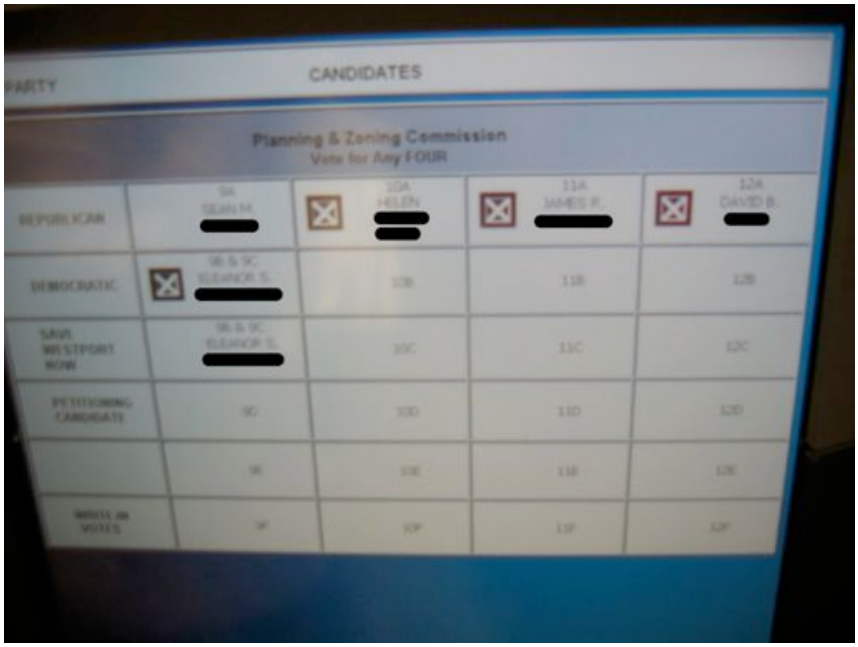


Figure 4: The ballot with unaltered candidates' names (before swapping)

Both of these appear in the GEMS database. Only the .rtf file is visible in the clear within the card contents though. The database label must be either encrypted or compressed with other data. The database label is used on the zero report and the final report, while the .rtf file is displayed on the screen and printed on the printed ballot.

Furthermore, we ran an election with two machines, each with a memory card, one of which was tampered in the aforementioned way (not displayed). After finishing the election, the results can be combined on the AV-TSx with no errors. That is, there is no check of whether the .xtr files match. Any votes for the blank spot would be assigned to the candidate that originally should have appeared there. The results can then be uploaded to GEMS with no errors.

Swapping Candidates: With the above observations, we then tried to swap two candidates with .rtf files of equal length along with their checksums. We again held a two machine election, swapping the .rtf files for one machine only. The slate presented by the untampered machine is given in Figure 4.

The tampered machine ran correctly, with the two candidates swapped on the screen compared to the untampered machine. Candidates 'DAVID B. XXXXX' and 'SEAN M XXXXX' are swapped (Figure 5).

We then voted twice for one of these candidates, candidate 'DAVID B. XXXXX', on each machine (with the original and tampered elections loaded correspondingly). The votes on the screen agreed with that on the printed VVPAT records (two for 'DAVID B. XXXXX') in both cases (see the scans of the records in Figure 6, Figure 7). Thus it appears that the election ran correctly and a voter can verify that the printed record indeed corresponds to the choices made on the screen. However, the final results on the tampered machine showed two votes for candidate 'SEAN M XXXXX' and zero for candidate 'DAVID B. XXXXX' (Figure 8). On untampered machine printed ballots and the results correpond to each other (Figure 7 and Figure 9 correpondingly). In this case we also were able to combine the results and send the tally to GEMS, with no errors. The results show two votes for each candidate 'DAVID B. XXXXX' and 'SEAN M XXXXX' correspondingly (Figure 10), even though during the election no votes were given to the candidate 'SEAN M

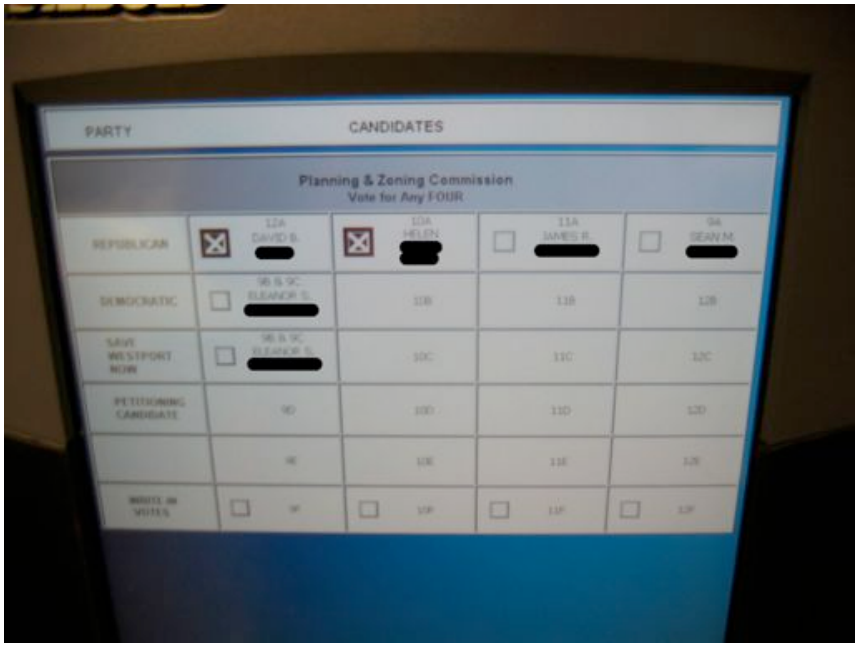


Figure 5: The ballot with swapped candidates' names

XXXXX'.

Conclusion: if an attacker has access to the memory card and two candidates have names of the same length, then the attacker can swap their votes on that machine. Note, the names themselves do not have to be the same size, but the .rtf files (these files contain formatting, such as spaces, newlines, and font information thus names of equal number of characters may still result in differently sized rtf files).

3.4 Completing the attack

Once all the changes have been made with the corresponding .xtr file the memory card is ready to be placed back. After this step, the AV-TSx terminal will be found by poll-workers in its expected pre-election state. The terminal will appear to be functioning normally for all operations during the election. The total time required to compromise the card is only a few minutes, depending on the dexterity of the attacker in picking the lock of the ballot box.

4 Conclusions

In this report, we performed a security analysis of the AV-TSx system and demonstrated two serious attacks against the integrity of the election process. It is important to point out the fact that we did not possess the source code for the voting terminal or GEMS, nor did we need any specialised equipment. Compromising a terminal takes a few minutes with the card and a hex editor. Most laptops these days are equipped with PCMCIA card readers, obviating the need for a special card reader. In the light of these attacks, it suggests that great caution is warranted before employing AV-TSx in an actual election. Some recommendations follow.

Figure 6: The votes represented on the printed ballot (altered case)

4.1 Improving the security of the AV-TSx

There have been several studies (e.g., [2, 9, 10]) that have specifically addressed the issue of designing e-voting systems and offering recommendations for improvement. Here, we point out the particular shortcomings of the AV-TSx terminal and identify aspects that need to be dealt with to obtain a secure and robust system.

Global Integrity Check. The memory card of the AV-TSx, a standard PCMCIA card, as discussed before, holds the election data, ballot box and the audit information. The major shortcoming in this regard is the lack of a global integrity check computed on the entire contents of the card. Our attacks were possible because of the lack of such a global check.

Modified Election Data Files and Integrity Checks. The .xtr file contains the names of the candidates in RTF format. Each .xtr file does have a 16 byte integrity check. A failed integrity check should under reasonable assumptions put the machine in an "insecure" state and have an alert presented. However, in the AV-TSx, a failed check of an RTF file simply makes that file not appear on the screen. A cryptographic check would not be effective if a failed check is not handled appropriately by the system.

Inconsistent File Usage. The candidate names that are printed for the voter verified paper trail are based on the same RTF file that is displayed to the voter, while the name printed for the final results is based on data from the .edb file. Because of this, a modified .xtr file may go undetected by initial testing by poll workers. The slate options displayed to voters should correspond exactly to the choices displayed on the final results.

Backdoor Files. Previous versions of the TS machine were susceptible to attacks through back door files [4]. Such files, if present on the memory card, gives the user unrestricted access to the terminal for debugging purposes. It is unclear whether there exist any backdoor files in use for the current AV-TSx terminal; further investigation would be necessary to make sure that such backdoors do not exist.

Figure 7: The votes represented on the printed ballot (unaltered case)

Figure 8: The results of the election held on a tampered machine

Figure 9: The results of the election held on unaltered machine

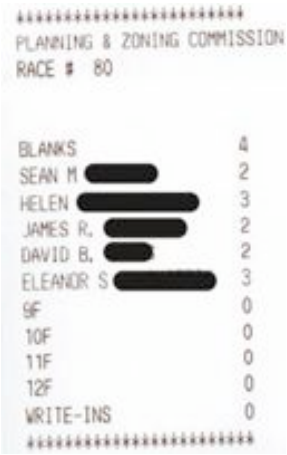


Figure 10: The results accumulated from both machines

Limited Software Accountability and Auditability. There is no (documented) way in which to examine the software (Operating System) currently installed on the machine.
Internal Flash Accumulator. The AV-TSx has the ability to accumulate voting results from several voting machines, each with their own memory card. This potentially may entail the possibility to "stuff" the ballot box prior to the machine being used. Further testing would be required to make sure that the terminal is not susceptible to this attack.

4.2 Safe Use Recommendations

In 2002, in the United States, the Help America Vote Act mandated that one handicapped accessible voting system be provided per polling place, which most jurisdictions have chosen to satisfy with the use of DRE voting machines, some switching entirely over to DRE. Thus, it is imperative to follow safe-use guidelines before deploying these terminals. Here, we outline a set of safe-use recommendations for the AV-TSx and extend to electronic voting machines in general.
Tamper Evidence. Any access point or storage media should be sealed in a tamper evident fashion. On the AV-TSx in particular, this includes the memory card PCMCIA slots, but on other systems could include network ports, PCMCIA slots, USB ports, serial ports, phone jacks etc. Indeed, the presence of a USB port on a voting machine with an operating system that has sufficient drivers could allow the connection of keyboards, flash memory devices, or other common devices in order to take control of the system.
Chain of Custody. Assuming that the machine is initially prepared by trustworthy individuals, the problem then is to ensure that the machine remains physically secure at all times prior to election day. As mentioned, the simple lock on the ballot box used for the AV-TSx machine was insufficient since it can be easily picked or a key obtained without much difficulty. The machines must also be secured after the election in case auditing is deemed necessary, especially in the absence of a paper trail to audit.
Removable Media. Voting terminals should perform a cryptographic integrity check on their removable devices. Without this capability, any removable device (e.g., a memory card) must be sealed and any removed device should be considered compromised.
Random Audits. Post-election random audits of the voting machines coupled with manual recounts would help identify faulty or compromised voting machines. In [3], a simple method for sampling precincts in an observable way is presented. There remains a concern, however, that the physical ballots that will be manually recounted are machine generated (as opposed to actually produced by the voters). This could open the door for biasing the election results as described in [1].

References

[1] Brennen Center Task Force on Voting System Security. *The machinery of democracy: Protecting elections in an electronic world, 2005.* Lawrence Norden, Chair. Brennen Center for Justice, NYU School of Law. www.brennancenter.org

[2] David Chaum, Peter Y. A. Ryan, Steve A. Schneider, A Practical Voter-Verifiable Election Scheme. ESORICS 2005, pp. 118-139.

[3] Arel Cordero, David Wagner, and David Dill. The Role of Dice in Election Audits – Extended Abstract, IAVoSS Workshop On Trustworthy Elections (WOTE 2006), June 29, 2006

[4] Ariel J. Feldman, J. Alex Halderman, and Edward W. Felten
Security Analysis of the Diebold AccuVote-TS Voting Machine
http://itpolicy.princeton.edu/voting/

[5] Harri Hursti, Diebold TSx Evaluation, Black Box Voting Project, May 11, 2006
http://www.blackboxvoting.org/BBVtsxstudy.pdf
http://www.bbvdocs.org/reports/BBVreportIIunredacted.pdf

[6] David Wagner, David Jefferson and Matt Bishop, Security Analysis of the Diebold AccuBasic Interpreter, Voting Systems Technology Assessment Advisory Board, University of California, Berkeley, February 14, 2006.

[7] Election Data Services Inc. 2006 Voting Equipment Study, February 6, 2006.
http://www.electiondataservices.com/EDSInc_VEStudy2006.pdf

[8] Diebold Election Systems FAQ http://www.diebold.com/dieboldes/faq.asp

[9] Rebecca Mercuri, A Better Ballot Box?, IEEE Spectrum, Volume 39, Number 10, October 2002.

[10] D. Molnar, T. Kohno, N. Sastry, and D. Wagner, Tamper-Evident, History-Independent, Subliminal-Free Data Structures on PROM Storage -or- How to Store Ballots on a Voting Machine. Extended abstract, in IEEE Security and Privacy, 2006.

[11] http://www.law.csuohio.edu/CERP/documents/CERP_A-E.pdf/

Electronic Voting Machines

A Summary Comparison of the Optical Scan (OS) and the Touch Screen (TS) Voting Terminals

This summary presents an impartial discussion of the two voting technologies in wide use as of this writing (2007): Optical Scan and Touch Screen technologies. The purpose of the presentation is to better the understanding of the pros and cons offered by these two technologies.

We begin by describing some advantages that the Optical Scan (OS) terminals offer over the direct-recording electronic (DRE) Touch Screen (TS) terminals.

Voter-Verified Paper Trail. A very important criterion in assessing voting technology is the provision of the Voter Verified Audit Trail (VVAT), that is a physical copy, for example, a paper record, of the actual vote that the voter verified before it was cast. VVAT is sometimes referred to as the Voter Verified Paper Ballot (VVPB). For the OS terminals this is the actual ballot sheet. For the TS terminals, this is the printed record that the voter can accept or reject. An obvious advantage with the OS terminals is that there is no need for an additional voter verifiable audit trail, as each hand-marked ballot is automatically "voter verified". The TS terminals on the other hand, do not provide a direct VVAT, as the system may need print several ballots that are rejected by the voter, following by the accepted ballot. This necessitates the need for automatic separation (and possibly mechanical or manual destruction) of the rejected ballots from the accepted ballots. This process is more cumbersome, and relies on a less direct correspondence between the casting of a vote and its verification by the voter. (We also note that not all commercial TS terminals provide printed VVPBs.)

Authentication Issues. Another noticeable distinction is in the actual voting process itself. Regardless of the technology used, each voter needs to be authenticated (say, by an authentication token) before s/he can be allowed to cast the vote. The OS technology simplifies this process by eliminating an authentication layer, i.e., by saving the time, expenses and the security considerations in making/securing voting cards, card readers, etc. When using the OS terminals the voter goes to the polling place, gets a ballot sheet

(which serves as an authentication token). When using the TS terminals the voter card plays the role of the authentication, necessitating further security measures and expense.

Although the OS terminals may save on the time/money spent on the voter cards and card readers they have the disadvantage of having to print and securely transport paper ballots ahead of time. The TS machines do not require many magnetic cards ready since they can be quickly cleared and reprogrammed. There is a tradeoff in practical terms: how many paper ballots to print vs. time spent creating, clearing, and reprogramming the magnetic cards. In most situations where OS is used, it is very hard to come to a polling place with multiple (possibly falsified) ballot sheets without getting noticed (walking in with a stack of ballots). The magnetic ID cards for TS systems may or may not be hard to reproduce, but it is yet one more security issue that the voter and poll workers need to deal with.

Private Time with the Voting Terminal. The next point of interest is that in the case of the OS terminals there is no need for private time for the voter with the voting terminal. The actual ballot-casting time is minimal, i.e., the time of interaction of the voter with the terminal is very small. Consequently, the OS machines can offer better throughput, i.e., they can serve more people per terminal since the voters can fill out the ballots simultaneously without having to use the machine while doing so. The OS machine can scan accept a voter's ballot in a few seconds, while the TS machine can serve one person at a time and this can take from a few minutes to a noticeable fraction of an hour per voter, depending on how complicated is the electronic ballot and how much the voter might need to make all necessary decisions. Thus, while the OS terminals are not only more efficient, but also more secure in the sense that since the voter has less time with the machine, there is also a lesser chance that an attacker can tamper with the voting terminal.

Hardware Failures During an Election. If an OS terminal fails, voting can continue, as the voters are still able to mark ballots that can later be manually or automatically counted. When a TS terminal fails, voting cannot continue, and there are non-trivial concerns of what votes have been recorded, and what ballots have been printed (if any).

We now overview some potential advantages that the TS technology can offer over the OS technology.

User Interface Issues: Multilingual Access and Voter with Disabilities. TS technology makes it easier to have user interfaces in multiple languages (e.g., English and Spanish, etc.). As with ATMs, the voter may simply select the language in which he or she wishes to vote at the start of the voting process. While ballots for OS systems can also be provided in multiple languages, there is expense associated with printing ballots in the required languages. TS terminals can also be designed for voters with limited manual dexterity and other physical disabilities, and can incorporate assistive technologies (e.g., headphones for the visually impaired), allowing the disabled to vote without forfeiting the anonymity of their vote.

Assistance with Vote Validation. The TS machines can provide better assistance to the voter in handling vote miscounting in comparison to the OS terminals. The OS machines can miscount when a voter marks the bubbles poorly or ambiguously. The TS systems provide feedback to the voters, showing unambiguously whether or not the terminal has received the selection made by the voter, and in case of malfunction, a voter may be able to observe that a TS terminal is not operating correctly. Both technologies provide adequate ways of dealing with overvotes. TS technology provides additional assistance in the case of undervotes, for example the terminal may ask a voter whether an apparent undervote is intentional.

No Need for Paper Ballots. Finally, with the TS voting systems there is the savings associated with not having to print paper ballots, and there is no risk of exhausting the supply of paper ballots.

Finally we note that this overview deals with the broad issues associated with the OS and TS technologies. The overview does not address the specific features, the security vulnerabilities, or the reliability issues associated with available commercial models of voting terminals.